Preconditioners for Krylov subspace methods: An overview

Received ; | Revised ; | Accepted

DOI: xxx/xxxx

ARTICLE TYPE

Preconditioners for Krylov subspace methods: An overview

John W. Pearson[1] | Jennifer Pestana*[2]

[1]School of Mathematics, The University of Edinburgh, Edinburgh, United Kingdom
[2]Department of Mathematics and Statistics, University of Strathclyde, Glasgow, United Kingdom

Correspondence
*Jennifer Pestana, Department of Mathematics and Statistics, University of Strathclyde, 16 Richmond Street, Glasgow, UK, G1 1XQ. Email: jennifer.pestana@strath.ac.uk

When simulating a mechanism from science or engineering, or an industrial process, one is frequently required to construct a mathematical model, and then resolve this model numerically. If accurate numerical solutions are necessary or desirable, this can involve solving large-scale systems of equations. One major class of solution methods is that of preconditioned iterative methods, involving preconditioners which are computationally cheap to apply whilst also capturing information contained in the linear system. In this article, we give a short survey of the field of preconditioning. We introduce a range of preconditioners for partial differential equations, followed by optimisation problems, before discussing preconditioners constructed with less standard objectives in mind.

KEYWORDS:
preconditioning, iterative method, Krylov subspace method, partial differential equations, optimisation

1 | INTRODUCTION

Solving large linear systems $\boldsymbol{Ax} = \boldsymbol{b}$, where $A \in \mathbb{C}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{C}^n$, is a mainstay of scientific computing. Such systems arise in many applications—perhaps most frequently the discretisation of partial differential equations and optimisation problems—and their solution can be extremely time-consuming. Moreover, for certain time-dependent, nonlinear, or inverse problems, for instance, multiple linear systems must be solved, so a reduction in the linear solve time results in a huge increase in the efficiency of the overall numerical scheme.

For the solution of a wide class of linear systems, the use of *direct methods* is both popular and powerful (see [1] for a survey of such methods for sparse systems). Here, a sequence of operations is performed on the matrix in order to generate a suitable factorisation, therefore returning a single estimate of the solution of the system. However, if the matrix in question becomes sufficiently large for a particular computer architecture, the storage and operation costs of a direct method may become excessive. In such a case it becomes valuable to apply *iterative methods*, where no such factorisation is required, in order to generate a sequence of approximate solutions. Such methods generally require matrix–vector multiplications at each iteration, but these may be performed without storing the entire matrix: one may work with individual blocks of the matrix, or indeed apply "matrix-free" approaches (see [2,3] for instance). Here, we focus specifically on Krylov subspace methods, which we detail in Section 2; for such methods A is indeed only accessed via matrix–vector multiplications. However, without the use of preconditioning (that we describe in greater detail in Section 2) the Krylov solver may not converge, or may take many iterations to return an acceptably accurate solution. In the best case, *preconditioned iterative methods* can substantially reduce the storage and operation costs as opposed to direct methods, while achieving greatly improved convergence properties that may often be proved *a priori*.

Two fundamental goals when devising a preconditioner can therefore be summarised as follows:

- ▷ Facilitating rapid convergence of an iterative method;
- ▷ Guaranteeing a reduction in terms of storage and computational operations as opposed to direct methods.

Accordingly, preconditioning allows for the solution of problems that cannot be tackled using either direct schemes or unpreconditioned iterative methods.

The design of such a preconditioner necessitates a trade-off, as one then needs to solve a linear system involving the preconditioner at least once per iteration. Hence, a preconditioner must be constructed such that it is cheap to apply its inverse operation to a given vector, while also capturing the characteristics of the original system in some sense. The second criterion will be explored further throughout this paper, as its interpretation depends on several factors including which iterative method is used.

There are already several excellent surveys of preconditioners in general, and for specific applications; we hope that this article complements these existing works. In particular, we mention the extensive surveys by Benzi[4] and Wathen[5], as well as the books by Bertaccini and Durastante[6] and Chen[7]. Further works surveying certain areas of preconditioning include[8–11]. A key structural difference in this paper is that we predominantly categorise by application area rather than by type of preconditioner. Throughout this text, we additionally point to surveys on individual topics.

We begin our discussion by describing the most widely used Krylov subspace methods in Section 2. We particularly emphasise the convergence theory available for these methods, since this can be used to determine preconditioners that fulfil the aims outlined above. Section 3 then details preconditioners appropriate for partial differential equation (PDE) problems, while Section 4 discusses preconditioners for linear systems that arise in optimisation. Although preconditioning typically focuses on ensuring that the preconditioned matrix has "favourable" properties, there are also other ways to precondition, and reasons for doing so, as we illustrate through selected examples in Section 5. Finally, we summarise our key messages in Section 6.

Notation: Throughout, we use bold lowercase letters, e.g., $\boldsymbol{x}$, to represent vectors, and uppercase letters, e.g., $\boldsymbol{M}$, for matrices. We denote by A a matrix that is being solved for and by P a preconditioner, except in the case that the block structure is important, when we use the calligraphic letters $\mathcal{A}$ and $\mathcal{P}$. In this case A can denote a sub-block of $\mathcal{A}$. A general matrix is represented by M. We denote the Hermitian transpose of $\boldsymbol{x}$ by $\boldsymbol{x}^H$ and the transpose by $\boldsymbol{x}^T$. If $M \in \mathbb{C}^{n\times n}$ is Hermitian positive definite then it defines a norm, which we denote by $\|\boldsymbol{x}\|_M := \sqrt{\boldsymbol{x}^H M \boldsymbol{x}}$. We will call a preconditioner *optimal* if it requires $O(n)$ floating point operations to apply, and if the number of iterations required to attain a certain accuracy can be bounded independently of n. Whenever we cite several references consecutively, such as[4,5,7], this is a possibly non-exhaustive list and should be read "see, e.g.,[4,5,7] and the references therein".

2 | KRYLOV SUBSPACE METHODS

Preconditioning has a surprisingly long history, with a crucial early reference to this term made by Alan Turing[12]. Another milestone was the paper on incomplete Cholesky preconditioners by Meijerink and van der Vorst[13], which clearly illustrated the potential of preconditioning to improve the efficiency of iterative solvers. In this section, to better understand the role of preconditioners, we provide a short introduction to Krylov subspace methods. This is by no means comprehensive, and we refer the interested reader to the excellent books by Greenbaum[14], Liesen and Strakoš[15], and Saad[16], and surveys by Eiermann and Ernst[17], Liesen and Tichý[18], and Simoncini and Szyld[19]. We note that although our focus here is on Krylov methods, the approaches to preconditioning described in this article can certainly be applied to other iterative algorithms.

Stationary iterative methods have been used at least since the time of Gauss, but over the last few decades they have been largely superseded by Krylov subspace methods, which approximate $\boldsymbol{x}$ from Krylov subspaces. Named after Aleksei Nikolaevich Krylov, who used these spaces to analyse oscillations of mechanical systems[20], Krylov subspaces take the form

$$\mathcal{K}_k(A, \boldsymbol{v}) := \text{span}\{\boldsymbol{v}, A\boldsymbol{v}, \dots, A^{(k-1)}\boldsymbol{v}\},$$

where $A \in \mathbb{C}^{n\times n}$ and $\boldsymbol{v} \in \mathbb{C}^n$.

The first Krylov subspace method for linear systems, namely the conjugate gradient method (CG), was proposed by Hestenes and Stiefel[21], with Lanczos also working on closely related topics in this period[22,23]. The conjugate gradient method must terminate in at most n iterations in exact arithmetic. Thus, although Hestenes and Stiefel[21] and Lanczos[23] refer to CG and the related Lanczos method as iterative solvers, over the next two decades mathematicians (in contrast to engineers and physicists) considered CG a direct method. However, in this capacity it was often outperformed by other approaches. It was only after John Reid's observation[24] that good approximations to $\boldsymbol{x}$ could be obtained in far fewer than n steps, that Krylov methods were again viewed as iterative methods[†]. Renewed interest followed, with new methods developed and analysed. For Hermitian matrices the

[†]This was also noted in[25], but its potential was not recognised until later.

most popular are probably MINRES and SYMMLQ[26] while for non-Hermitian matrices such methods include GMRES[27], FOM[28], LSQR[29], LSMR[30], and families of methods based on QMR[31], BiCG[32] and BiCGStab[33], and IDR[34,35].

Krylov subspace methods for linear systems start from an initial guess $\boldsymbol{x}_0$ (often the zero vector), compute the initial residual $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, and then (typically) select iterates $\boldsymbol{x}_1$, $\boldsymbol{x}_2$,... such that

$$\boldsymbol{x}_k - \boldsymbol{x}_0 \in \mathcal{K}_k(A, \boldsymbol{r}_0), \tag{1}$$

thereby projecting the original problem to one of much smaller dimension. This approach is naturally linked to polynomial approximation, since $\boldsymbol{x}_k - \boldsymbol{x}_0 \in \mathcal{K}_k(A, \boldsymbol{r}_0) \Leftrightarrow \boldsymbol{x}_k = \boldsymbol{x}_0 + q_{k-1}(A)\boldsymbol{r}_0$ for some $q_{k-1} \in \Pi_{k-1}$ with Π_k denoting the set of polynomials of degree at most k. Accordingly, the kth residual satisfies

$$\boldsymbol{r}_k = \boldsymbol{b} - A\boldsymbol{x}_k = (I - Aq_k(A))\boldsymbol{r}_0 = p_k(A)\boldsymbol{r}_0, \quad p_k \in \Pi_k,\ p(0) = 1.$$

We have not yet described how iterates are chosen from these Krylov subspaces, and indeed this choice distinguishes different methods. We will not discuss this in detail here, but will instead focus on the available convergence theory for these methods.

2.1 | Hermitian problems

We first consider the conjugate gradient method for Hermitian positive definite A, which is perhaps the best understood Krylov method. CG chooses $\boldsymbol{x}_k$ according to (1) such that $\|\boldsymbol{e}_k\|_A$ is minimised, where $\boldsymbol{e}_k = \boldsymbol{x} - \boldsymbol{x}_k$. It requires only short-term recurrences so that the computational work at each iteration is fixed. Crucially for preconditioning we can bound $\|\boldsymbol{e}_k\|_A$ as follows:

$$\frac{\|\boldsymbol{e}_k\|_A}{\|\boldsymbol{e}_0\|_A} \leq \min_{\substack{p \in \Pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|,$$

where $\sigma(A)$ is the spectrum of A. This bound is sharp, in the sense that for every k there is a (possibly different) initial guess $\boldsymbol{x}_0$ for which it is attained. That said, given a specific initial residual, i.e., a specific right-hand side and initial guess, the convergence rate may be much faster than is predicted by the above bound. We refer to the book by Liesen and Strakoš[15, Section 5.6] and the references therein for a careful CG convergence analysis.

When A is Hermitian but indefinite, we can instead apply MINRES, which also has fixed work per step and for which an analogous convergence bound can be derived. MINRES chooses $\boldsymbol{x}_k$ such that $\|\boldsymbol{r}_k\|_2$ is minimised subject to (1), and in this case

$$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{r}_0\|_2} \leq \min_{\substack{p \in \Pi_k \\ p(0)=1}} \max_{\lambda \in \sigma(A)} |p(\lambda)|. \tag{2}$$

This bound is descriptive in the same sense as the CG bound mentioned above[18], but the eigenvalues of A may also be negative.

We see from the above bounds that convergence of CG and MINRES is strongly related to the spectral properties of A. Since these are governed by the problem at hand, it is often desirable to improve the convergence rate by solving an equivalent linear system with "better" spectral properties, i.e., by preconditioning. To preserve symmetry, for CG and MINRES we precondition symmetrically, i.e., one may instead consider the formalism

$$C^{-1}AC^{-H}\boldsymbol{y} = C^{-1}\boldsymbol{b}, \quad \boldsymbol{x} = C^{-H}\boldsymbol{y},$$

where $C \in \mathbb{C}^{n \times n}$ is nonsingular. For this preconditioned system, the spectral properties of $C^{-1}AC^{-H}$ become important, and by a careful choice of C it is hoped that convergence will be faster. We note that by rewriting the CG or MINRES algorithm it is possible to avoid computations with C and C^H and work only with the symmetric positive definite matrix $P = CC^H$. However, the preconditioner for CG and MINRES must be Hermitian positive definite in general.

2.2 | Non-Hermitian problems

GMRES is an extension of MINRES to non-Hermitian systems, and because it minimises $\|\boldsymbol{r}_k\|_2$ it has the most complete convergence theory out of solvers for such systems. If A is normal (2) holds, although now the eigenvalues of A may be complex, and this makes the convergence of GMRES more challenging to understand. However, when A is far from normal, no convergence bound is universally descriptive[36], and the spectrum alone cannot characterise the convergence rate of GMRES[37–40]. A major disadvantage of GMRES is that the computational work grows with each iteration, and one option is to restart GMRES[16, Chapter 6]. Alternatively we may use a method based on bi-orthogonalisation, such as QMR, BiCGStab, or IDR. However, none of these methods has a tractable optimality property, and so convergence results are extremely limited.

Thus for non-Hermitian, particularly nonnormal, problems preconditioning is heuristically motivated although, as we shall describe, this has not prevented the development of many effective preconditioners for a range of such systems. We note that for non-Hermitian problems it is possible to perform

- ▷ left preconditioning, i.e., solve $P^{-1}A\boldsymbol{x} = P^{-1}\boldsymbol{b}$;
- ▷ right preconditioning, i.e., solve $AP^{-1}\boldsymbol{y} = \boldsymbol{b}$, with $\boldsymbol{x} = P^{-1}\boldsymbol{y}$;
- ▷ or split preconditioning, i.e., solve $P_1^{-1}AP_2^{-1}\boldsymbol{y} = P_1^{-1}\boldsymbol{b}$, with $\boldsymbol{x} = P_2^{-1}\boldsymbol{y}$.

A key restriction is that the preconditioner must represent some linear operator. If this is not the case a flexible Krylov subspace should be used [16, Section 9.4].

We end this section by remarking that it is always possible to symmetrise a problem by, e.g., solving the normal equations $A^HA\boldsymbol{x} = A^H\boldsymbol{b}$. CG or MINRES can be applied to this symmetrised system, although in practice the more numerically stable but mathematically equivalent LSQR (for CG) or LSMR (for MINRES) is used. In this case we again recover descriptive convergence theory. On the other hand, convergence rates for problems solved using the normal equations are often significantly worse than for the original linear system, and preconditioning A^HA (or AA^H) is often challenging; indeed it is possible that P^HP is an arbitrarily poor preconditioner for A^HA even if P is a suitable preconditioner for A [41].

3 | PRECONDITIONERS FOR PARTIAL DIFFERENTIAL EQUATIONS AND RELATED PROBLEMS

Often, linear systems arise upon the discretisation of an infinite-dimensional problem, e.g., differential and partial differential equations, or integral equations. Here we focus on second-order partial differential equations, but some of the underlying themes and principles can be applied to, or adapted for, discretisations of other infinite-dimensional problems.

An important point is that solving the linear system is not an end in itself; instead, the overarching goal is to approximate the solution to the infinite-dimensional problem. The accuracy of this solution is affected by the discretisation error as well as the error in terminating the iterative solver (the algebraic error). The interplay between these errors [42,43], and their effect on the stopping criteria [44–46], should be considered, although this is typically a non-trivial task.

The convergence rates of iterative methods for discretised problems are influenced both by the properties of the infinite-dimensional problem and the choice of discretisation. Knowledge of the properties of the underlying PDE (particularly the weak form) has been exploited to develop so-called "operator" preconditioners [47–52] [53, Chapter 2]; for an overview of these, see [54, Chapter 4] [55]. The recent paper [56] also treats operator preconditioners in infinite- and finite-dimensional settings and describes an abstract framework for operator preconditioners in terms of a splitting of a Hilbert space into a finite number of subspaces. Other preconditioners that, at least in their original forms, are based on knowledge of the underlying differential equation include multigrid and domain decomposition, which we discuss in more detail below.

Discretisation choices also affect the properties of the coefficient matrix, and hence the convergence rate of the Krylov subspace method. In particular, PDE problems require global transfer of information but popular discretisations, e.g., finite element, finite difference, and finite volume methods, lead to sparse matrices that transfer information only locally. Preconditioning PDE problems can be viewed as transforming to a more appropriate basis [54, Chapter 8]. This perspective has also been used more explicitly in hierarchical basis preconditioning [57,58] (see also the survey by Xu [59]), and cardinal basis preconditioning for radial basis discretisations [60,61], which were originally introduced in the context of interpolation [62].

In this section we discuss preconditioners for some prototypical PDEs; these PDEs either arise on their own or, increasingly commonly, as key components of more complicated, possibly nonlinear PDE problems. These include the Poisson, Helmholtz, and convection–diffusion equations, which represent problems involving a single scalar-valued variable, and the Stokes and Navier–Stokes equations, which are examples of systems of PDEs. We also discuss specialised methods for Toeplitz matrices, time-dependent problems, and problems with heterogeneous coefficients.

3.1 | Single PDEs

3.1.1 | Poisson problems

The first equation we consider is the generalised Poisson problem: find u such that

$$-\nabla \cdot (\alpha \nabla u) = f \quad \text{in } \Omega \tag{3}$$

subject to appropriate boundary conditions, where α is a sufficiently smooth positive function that is both bounded above and uniformly bounded away from zero on Ω, and $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$ is a bounded connected domain with piecewise smooth boundary $\partial\Omega$. (The standard Poisson problem is recovered when $\alpha \equiv 1$.) Equation (3) is elliptic and self-adjoint, and arises in many applications in e.g., chemistry, astrophysics, fluid dynamics, solid mechanics, electromagnetics, and image processing. It is also a subproblem of more complicated PDEs, such as the Navier–Stokes equations. Discretisation by standard numerical methods leads to a linear system $A\boldsymbol{x} = \boldsymbol{b}$, where $A \in \mathbb{R}^{n\times n}$ is symmetric and at least positive semidefinite.

When (3) has separable coefficients on rectangular domains, it may be discretised by finite difference methods and then solved by tailored direct methods known as fast Poisson solvers[63–66]. However, for more complicated problems, domains, meshes, or discretisations, the preconditioned conjugate gradient method is usually the method of choice. In these cases, fast Poisson solvers may be used as preconditioners, although their effectiveness deteriorates as the complexity of the problem increases. These fast Poisson solver preconditioners can also be viewed as operator preconditioners that approximate $(-\nabla^2)^{-1}$, see, e.g., [55]. Recently, it was shown that when Laplace preconditioners are applied to finite element discretisations of (3) with non-constant α, the eigenvalues of the preconditioned matrix are approximated by nodal values of α[67].

Multigrid

More robust preconditioners for elliptic, and indeed hyperbolic and parabolic, PDEs include multigrid methods, which are iterative solvers in their own right. Standard multigrid methods are based on the observation that when stationary iterations such as Jacobi or Gauss–Seidel are applied to problems from PDEs, in many cases they quickly reduce high-frequency components of the error but are less effective at removing smooth, low-frequency, components. Multigrid exploits this, and a sequence of smaller linear systems, to accelerate the convergence of these stationary iterations.

Multigrid is more easily understood by first considering a two-grid scheme. Within this two-grid iteration, a few steps of a stationary iterative method result in the approximation $\hat{\boldsymbol{x}}$ to $\boldsymbol{x}$, with smooth error $\boldsymbol{e} = \boldsymbol{x} - \hat{\boldsymbol{x}}$. We then improve $\hat{\boldsymbol{x}}$ by approximately solving the associated residual equation $A\boldsymbol{e} = \boldsymbol{r}$. Specifically, since $\boldsymbol{e}$, hence $\boldsymbol{r}$, are smooth they are well represented by a smaller (coarse grid) problem $A_c\boldsymbol{e}_c = \boldsymbol{r}_c$ that is more easily solved. Projecting $\boldsymbol{e}_c$ back to the fine grid then gives $\hat{\boldsymbol{e}} \approx \boldsymbol{e}$, and we obtain the improved approximation $\boldsymbol{x} \approx \hat{\boldsymbol{x}} + \hat{\boldsymbol{e}}$. Applying the stationary iterative method again at this stage (to post-smooth) is optional.

In multigrid, the process of smoothing and projecting is repeated recursively until we reach a problem that is small enough to be solved by a direct method; to obtain $\hat{\boldsymbol{e}}$ we then interpolate this approximate solution on successively finer grids by traversing the grid hierarchy in the opposite direction. The motivation for this repeated grid transfer is that low-frequency error components for the original problem become high-frequency components when represented on this coarse grid, which can again be smoothed by the stationary iteration. If a single two-grid iteration is performed at each recursive level we obtain a V-cycle; if two such iterations are performed on each recursive level we instead obtain a W-cycle, see, e.g., [68, Section 2.4].

Multigrid ideas appeared in early papers[69,70], but the seminal work by Brandt[71] is widely regarded as the foundation of modern multigrid methods. These solvers were geometric multigrid (GMG) methods, using a sequence of meshes for Ω. GMG is convenient for simpler problems and structured grids for which interpolation and restriction of the error are more straightforward.

For complex domains, unstructured meshes, and/or heterogeneous coefficients, it can be challenging to develop effective GMG methods. To overcome this, algebraic multigrid (AMG) methods were developed from the 1980s onwards[72–76]. The key distinction is that coarsening is performed automatically using only matrix entries, in a way that mimics GMG methods. AMG is now widely used, not only for problems that are challenging for GMG, but also to avoid programming problem-specific solvers.

Multigrid methods have been the subject of much research over recent decades and further details can be found in, e.g., [68,77–80]. (Linear) multigrid is itself a (complicated) stationary iterative method; for many simpler problems it is also an optimal solver and in these cases it can be used on its own. For more complicated problems multigrid may converge slowly, if at all, because some error modes are not adequately damped. In this situation it is much more effective to use a fixed number of multigrid iterations as a preconditioner for the conjugate gradient method, provided the multigrid method represents a symmetric positive definite preconditioner, e.g., a V-cycle or W-cycle with smoothing during the interpolation phase that is symmetric with respect to the smoothing used when coarsening. Note that using a variable number of multigrid iterations, or an otherwise nonlinear method, results in a nonlinear preconditioner that should not be used with standard Krylov subspace methods.

Domain decomposition

Domain decomposition methods also replace the original Poisson linear system by smaller problems but, in contrast to multigrid methods, they divide the domain into subregions and solve (3) on each. Doing so requires the imposition of artificial boundary conditions, called transmission conditions, on subdomain boundaries (except where the subdomain boundary is part of

$\partial\Omega$). These transmission conditions allow the transfer of information between subdomains. Whilst subdomain solves originally corresponded to physical regions of Ω, it is also possible to specify subdomains using subsets of rows and columns of A.

The first domain decomposition method was proposed by Schwarz in 1870[81] to prove existence and uniqueness of solutions to (3) on complicated domains by dividing them into two simpler subdomains. This alternating Schwarz method solved over each subdomain in turn, using the most up-to-date solution information to impose Dirichlet boundary conditions. Multiplicative Schwarz methods generalise this idea, but additive Schwarz and related methods[82,83] are generally preferred; one reason for this is that they can more easily parallelised. In particular, the popular restricted additive Schwarz method adds the solution to each overlapping subdomain problem, using a partition of unity in overlap regions.

Additive Schwarz, multiplicative Schwarz, and related methods are also stationary iterative schemes that can be used as standalone solvers provided they converge[84,85]. However, local transfer of information means that their convergence rates can be slow, and so they are usually supplemented by some sort of global coarse grid solve. This global solve may additionally deal with modes that impede convergence[86–88].

Like multigrid, for complex problems domain decomposition methods are often much more effective as preconditioners. There are many such preconditioners available that differ in their choice of subdomains, overlap, transmission conditions, and coarse grid solves. Domain decomposition methods remain an active research area, and we recommend several books and surveys[89–95].

3.1.2 | Helmholtz problems

A seemingly small modification of the Poisson equation (3) gives rise to the Helmholtz problem

$$-\nabla^2 u - k^2 u = f \text{ in } \Omega, \tag{4}$$

again augmented with appropriate boundary conditions, with k real-valued and $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, a bounded connected domain with piecewise smooth boundary $\partial\Omega$. This important PDE features in, e.g., wave propagation problems and quantum mechanics, with oscillatory problems—that are characterised by moderate to large values of k—typically of interest.

Once discretised, (4) leads to a linear system $A\boldsymbol{x} = \boldsymbol{b}$, where $A = K - k^2 M \in \mathbb{R}^{n\times n}$, with K the discretised negative Laplacian and M, e.g., the identity matrix or a finite element mass matrix. A number of issues make the solution of this system challenging. First, complicated boundary conditions are often required for well-posedness and/or to mimic an infinite domain, and these can make A non-Hermitian, and possibly complex. Second, fine meshes are usually required to resolve the oscillatory solution and to avoid the "pollution effect", i.e., numerical dispersion; this results in extremely large linear systems. Finally, even if A is Hermitian, as we consider here, it is almost always indefinite. In fact, A is singular if k^2 is an eigenvalue of K, i.e., in the presence of resonances, although, e.g., MINRES can be applied to singular but consistent systems.

Standard preconditioners for elliptic problems generally perform poorly on discretisations of the indefinite Helmholtz problem. Multigrid preconditioners may struggle to smooth errors, and the oscillatory nature of the problem limits the sizes of coarse grids[96]; however, tailored multigrid[97–99] and multilevel[100] methods attempt to remedy these issues. Domain decomposition methods also require careful treatment, e.g., a sufficiently fine coarse grid[101], special coarse grid solves[102,103], and/or modified transmission conditions[104–106].

Instead of approximating A directly, it is possible to approximate a 'nicer' elliptic operator such as the negative Laplace operator[107] or the shifted operator[108] $-\nabla^2 + k^2 I$ by standard multilevel approximations. Unfortunately, the performance of these preconditioners deteriorates as k increases, as might be expected. More general "shifted Laplace" preconditioners based on $-\nabla^2 + \gamma I$, with γ complex, typically perform better[109–111] but an appropriate choice of shift is important for efficiency[112,113]; small shifts reduce the number of Krylov subspace iterations but make the preconditioner itself difficult to approximate well. An additional issue is that a complex shift makes the problem complex and non-Hermitian. It is possible to instead solve a real block system, but this doubles the problem dimension and two elliptic subproblems must be solved per iteration[114].

Sweeping preconditioners have also received much attention of late. These incomplete (block) factorisation preconditioners sweep through the variables in layers, using information about the underlying PDE to approximate Schur complements[115,116] (see the survey[117]). Recently, a different approach was proposed that utilises contour integrals to project the spectrum of A, transforming the original problem to one that is easier to solve[118]. Surveys of Helmholtz preconditioners can be found in[119,120].

The key point is that the preconditioners of choice for the Helmholtz problem are not necessarily the same as for the Poisson problem. This is essentially because features of the PDE (oscillations, and loss of ellipticity) strongly influence properties of A, which must then be accounted for when designing preconditioners.

3.1.3 | Convection–diffusion problems

Non-self-adjoint PDEs lead to linear systems with non-Hermitian coefficient matrices that must be solved by one of the non-symmetric Krylov methods introduced in Section 2. Among the best-studied PDEs of this type is the convection–diffusion equation

$$-\nabla^2 u + \boldsymbol{w} \cdot \nabla u = f$$

in $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$, with appropriate boundary conditions. One reason for this interest is that the Krylov subspace residual norms $\|\boldsymbol{r}_k\|_2$ for unpreconditioned linear systems arising from convection–diffusion problems are often characterised by an initial period of stagnation, during which the residual norm reduction is extremely slow, followed by a second, faster convergence phase. This two-phase behaviour is challenging to characterise using convergence bounds, and may be poorly described by the eigenvalues and eigenvectors alone [15, Section 5.7] [121, Chapter 26]. (We stress, however, that there are many linear systems with nonnormal coefficient matrices for which eigenvalues and eigenvectors do describe the convergence behaviour.)

Preconditioning is usually vital to reduce or remove this stagnation, and accelerate convergence. However, the nonnormality of the coefficient matrix makes it difficult to know *a priori* what should be achieved by a preconditioner. Despite this, many successful preconditioners have been proposed. For mildly convective problems, these may be based on the symmetric part of the coefficient matrix, i.e., the part associated with the negative Laplace operator [122,123]. For convection-dominated problems, however, it is necessary to incorporate the nonsymmetric part. For example, the negative Laplace preconditioner can be replaced by stationary iterative methods that split the coefficient matrix into its Hermitian and skew-Hermitian parts [124,125].

Even the (block) Gauss–Seidel iteration may make for an effective preconditioner if variables are ordered in the direction of the flow [126,127] [128, Chapter 7], and reordering is at the heart of incomplete factorisation [129] and splitting preconditioners [130,131] for the convection–diffusion problem. For more complicated flows, particularly recirculation problems, it is challenging to order the variables well, so many preconditioners are based on simplifying the convection–diffusion operator [132,133]. A completely different option is to employ a matrix-equation solver for separable-coefficient problems as a preconditioner [134,135].

Multigrid preconditioners can also be be effective, although advection causes difficulties that are not present for the Poisson problem. Gauss–Seidel smoothers can work well, but again for best effect the variables should be ordered according to the flow direction, or multiple sweeps in different directions should be used [80,128]. More generally, careful ordering of unknowns can improve multigrid preconditioners [136, Section 10.4.3] [137,138]. In geometric multigrid methods, it is also important to avoid oscillations resulting from unstable coarse grid discretisations, which require more smoothing steps to remove. Options for dealing with oscillations include semi-coarsening [139,140], matrix-dependent transfer operators [68, Section 7.7.5] [80, Section 5.4], upwinding [137,141], or rediscretising the coarse grid operator at each level using an appropriate stable discretisation [142]. Robust multigrid methods for the Navier–Stokes equations also tackle the difficult problem of building a scalable preconditioner for a convection–diffusion operator [143,144]. Algebraic multigrid methods can make good preconditioners for convection–diffusion problems [145–147] by automatically identifying layers and/or semi-coarsening. On the other hand, it may take many levels to reach an acceptably small 'coarse grid' problem [147].

3.2 | Systems of PDEs

Systems of PDEs can arise for a number of reasons, the most obvious cause being interactions between different components, e.g., in fluid–structure interactions or the magneto–hydrodynamics equations. Such systems also naturally occur when there are interactions between different quantities, e.g., the pressure and velocity of a fluid in the Navier–Stokes equations. Alternatively, it may be more convenient to write a single high-order equation as a system of lower-order equations; a classical example is the biharmonic equation, which can be reformulated in terms of two second-order equations[‡]. Systems may also arise when it is helpful to separate different components of the discretised problem, e.g., in finite element discretisations where it can be advantageous to group variables corresponding to different types of basis functions [149]. Alternatively, it may be advantageous to distinguish equations corresponding to boundary nodes, which may exhibit different behaviour, or may be discretised differently [150].

3.2.1 | Navier–Stokes problems

We start with coefficient matrices arising from general PDE systems. As an example we consider the primitive variables formulation of the steady incompressible Navier–Stokes equations, which couple the velocity $\boldsymbol{u}$ and pressure p of a fluid via the

[‡] On the other hand, it may instead be advantageous to move from a system of PDEs to a single higher-order PDE when developing preconditioners [148].

following equations:

$$\begin{aligned} -\nu\nabla^2\boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p &= \boldsymbol{f}, \\ -\nabla\cdot\boldsymbol{u} &= 0, \end{aligned} \tag{5}$$

subject to appropriate boundary conditions, on a bounded connected domain $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$ [128, Chapter 8]. The parameter $\nu > 0$ here represents the kinematic viscosity.

Since the Navier–Stokes equations are nonlinear, linearisation is required, with popular choices including Newton and Picard iteration. We consider Picard iteration here, which gives rise to a sequence of *Oseen problems*. Care is also needed when discretising (5) to ensure that the resulting method is stable; this is related to the inf-sup Ladyzhenskaya–Babuška–Brezzi (LBB) conditions [128, Chapter 3]. If a stable discretisation is used, such as an LBB-stable finite element pair or the marker-and-cell finite difference scheme, then each Oseen problem is a *saddle point system* of the form

$$\underbrace{\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{c} \end{bmatrix}. \tag{6}$$

Here, $A \in \mathbb{R}^{n\times n}$ contains the diffusive and convective terms while $B \in \mathbb{R}^{m\times n}$, $m \leq n$ represents the discrete negative divergence. We assume for simplicity that A is invertible and that B is full rank. (For conditions on the invertibility of $\mathcal{A}$ see [151, Section 3.2].) Note that the $(2,2)$ block in (6) may be replaced by a negative semi-definite matrix to stabilise a discretisation, and many of the preconditioners we discuss can be adapted to deal with this structure.

One widely-used option is to precondition $\mathcal{A}$ monolithically, e.g., via a multilevel method such as multigrid or domain decomposition [151, Section 11]. Alternatively, given the saddle point form of the Oseen equations, a number of preconditioners described in Section 4 can also be applied to (6). Provided ν is large enough, preconditioners designed for the Stokes equations (cf. Section 3.2.2) might also be suitable.

Yet another option is to exploit the block structure of $\mathcal{A}$ when designing preconditioners. For problems in which different blocks are associated with different physical quantities, or different PDEs, block preconditioners are often known as *physics-based preconditioners*. These block preconditioners may take advantage of the specific problem structure by, e.g., separating different spatial dimensions [152,153]. Alternatively, many popular preconditioners are based on the following "ideal" choices that exploit only the saddle point structure [154]:

$$\mathcal{P}_u = \begin{bmatrix} A & B^T \\ 0 & \pm S \end{bmatrix} \quad \text{or} \quad \mathcal{P}_l = \begin{bmatrix} A & 0 \\ B & \pm S \end{bmatrix},$$

where $S = BA^{-1}B^T$ denotes the (negative) Schur complement. Related block diagonal preconditioners can also be applied, but we defer discussion of these to Section 3.2.2. Both $\mathcal{P}_u$ and $\mathcal{P}_l$ are ideal, since a method like preconditioned GMRES will converge in at most two iterations with either choice. On the other hand, the size of A makes forming S and applying $\mathcal{P}_u$ or $\mathcal{P}_l$ too costly in general, and so many practical preconditioners are based on approximating A and S [128,155–157]. We stress that this block triangular approach can be applied to any saddle point problem, e.g., porous media flow [158,159].

3.2.2 | Stokes problems

When advective inertial forces are small compared with viscous forces in (5) we can approximate the Navier–Stokes equations by the linear Stokes equations:

$$\begin{aligned} -\nabla^2\boldsymbol{u} + \nabla p &= \boldsymbol{f}, \\ -\nabla\cdot\boldsymbol{u} &= 0, \end{aligned} \tag{7}$$

posed within $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, with appropriate boundary conditions.

Discretisation of these equations by, e.g., an LLB-stable finite element pair or the marker-and-cell finite difference method will lead to a saddle point problem of the form (6) in which $\mathcal{A}$ is symmetric. Specifically, $A \in \mathbb{R}^{n\times n}$ is a discrete representation of the negative vector Laplacian and $B \in \mathbb{R}^{m\times n}$, $m \leq n$, represents the discrete negative divergence (cf Section 3.2.1). We assume for simplicity that A is symmetric positive definite and that B has full rank although, as for the Navier–Stokes equations, these conditions may be relaxed [151].

The matrix $\mathcal{A}$ is symmetric but indefinite, making it somewhat challenging to solve (6). It is possible to instead solve the symmetric positive (semi-)definite Schur complement system

$$S\boldsymbol{y} = BA^{-1}B^T\boldsymbol{y} = BA^{-1}\boldsymbol{b} - \boldsymbol{c}, \tag{8}$$

where $S = BA^{-1}B^T$ is again the (negative) Schur complement (cf. Section 4), but since our focus here is on dealing with systems of PDEs we instead focus on solving the full system (6).

Given the similarity of the discretised Stokes equations with the Oseen problem, it is not surprising that many of the preconditioners discussed there can be adapted for the Stokes equations. However, since $\mathcal{A}$ is symmetric, it may be desirable to use a symmetric Krylov method, such as preconditioned MINRES, which requires a symmetric positive definite preconditioner. Doing so precludes certain monolithic and block triangular preconditioners.

When a positive definite preconditioner is required, a natural choice for (6) is a block diagonal preconditioner, such as[154,160]

$$\mathcal{P} = \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix}, \tag{9}$$

for which preconditioned MINRES converges (in exact arithmetic) in at most three steps. On the other hand, as for the block triangular preconditioners for the Oseen problem, in practice A and S must be approximated. Since A is a vector Laplacian, methods for the Poisson equation (see Section 3.1.1) can be used in its approximation. Thus, as for many block problems, the challenge lies in approximating S. Luckily, analysis (see, e.g.,[128, Chapter 4]) shows that when an LBB-stable finite element pair is used, a good approximation to S is the mass matrix on the pressure space, Q, or an approximation, e.g., its diagonal, a lumped version, or a fixed number of steps of Chebyshev semi-iteration[161–163]. Additionally,

$$\mathcal{P} = \begin{bmatrix} A & 0 \\ 0 & Q \end{bmatrix}$$

is an operator preconditioner[55]. For the marker-and-cell finite difference scheme, life is even easier since we can replace S by the identity matrix. We note that the preconditioner (9) has been generalised to certain problems containing more blocks[164–167], and that in general block diagonal preconditioning is a popular choice for symmetric saddle point problems[168–171]. The operator preconditioning approach can also be more widely employed; for more examples, see[55].

One aspect that is not often taken into account by block preconditioners is the appropriate scaling factors for the different blocks. Scaling can have an effect on the number of iterations needed for convergence[172,173]. More fundamentally, different equations are expressed in terms of different physical units, and these units may not be faithfully represented in the preconditioned system without scaling the blocks by relevant physical parameters, e.g., the viscosity in the Stokes equations[174]. This need becomes more obvious when dealing with varying parameters, as in two-phase flow[155,175].

3.3 | Toeplitz matrices

When constant-coefficient PDEs are discretised on uniform grids on simple domains, *Toeplitz* and *multilevel Toeplitz* matrices may arise. In this case, the highly structured nature of these matrices can be exploited to construct fast solvers. Although these structures have been utilised in the context of PDEs for some time[176,177], recent interest in the discretisation of fractional differential equations (FDEs) has proved a new source of (multilevel) Toeplitz problems[178,179]. This is because nonlocal fractional operators make the solution of the required linear systems extremely challenging unless tailored approaches are employed.

When one-dimensional linear constant-coefficient ordinary differential equations with Dirichlet boundary conditions are discretised on a uniform grid by, e.g., standard finite difference methods, the resulting coefficient matrix is Toeplitz, i.e., of the form

$$A = \begin{bmatrix} a_0 & a_{-1} & \dots & a_{-n+2} & a_{-n+1} \\ a_1 & a_0 & a_{-1} & & a_{-n+2} \\ \vdots & a_1 & a_0 & \ddots & \vdots \\ a_{n-2} & & \ddots & \ddots & a_{-1} \\ a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \end{bmatrix}.$$

If the boundary conditions are instead periodic then A is not only Toeplitz, but *circulant*, i.e., $(A)_{ij} = (A)_{(i-j) \bmod n}$. In two or more spatial dimensions the discretisation of PDEs no longer leads to Toeplitz matrices in general. However, when linear constant-coefficient problems are discretised on uniform tensor-product grids we may obtain multilevel Toeplitz matrices. The best known of these are block Toeplitz with Toeplitz block (BTTB) matrices for two-dimensional problems.

Although for ordinary or partial differential equations the resulting (multilevel) Toeplitz matrices are typically sparse, the non-locality of fractional operators means that matrices arising from FDE applications are dense. It is unusual for Krylov subspace methods to be competitive with direct methods for dense matrices, but here we can compute matrix–vector products in $O(n\log(n))$ operations by embedding A in a $2n\times 2n$ circulant matrix and utilising the fast Fourier transform (FFT) [180, Section 4.8.2].

Toeplitz matrices from differential equations are usually generated by a symbol, f, in the sense that the matrix entries are its Fourier coefficients, i.e.,

$$a_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)\mathrm{e}^{-\mathrm{i}k\theta}\,\mathrm{d}\theta.$$

(Multilevel Toeplitz matrices lead to multivariate symbols.) This connection with the symbol is extremely powerful, and many properties of A can be deduced from f. For example, if $f \in L^1((-\pi,\pi))$ is real-valued then A must be Hermitian and

$$\operatorname*{essinf}_{\theta\in(-\pi,\pi)} f(\theta) \le \lambda_{\min}(A) \le \lambda_{\max}(A) \le \operatorname*{esssup}_{\theta\in(-\pi,\pi)} f(\theta),$$

where the inequalities are strict if f is not constant almost everywhere. Asymptotic (in n) eigenvalue or singular value distributions of A can also be determined from f [181–184]; since the symbols of differential and integral operators typically have zeros, these results clearly illustrate why A is often extremely ill-conditioned for large n (as is well known).

Noticing that the inverse of a circulant matrix can also be applied to a vector in $O(n\log(n))$ operations using the FFT, Strang [185] and Olkin [186] proposed certain circulant preconditioners. This led to the development of a number of other preconditioners based on circulants [187,188], fast transforms [189,190], banded Toeplitz matrices [191,192], and multigrid [193] (see the survey by Chan and Ng [194] and the books [195,196]). For multilevel Toeplitz matrices, multilevel circulant preconditioners are not optimal since iteration numbers increase with n [197,198] but e.g., multigrid methods can give mesh-independent convergence rates [199], and semi-circulant preconditioners can be effective for certain PDE problems [176,177].

The vast majority of preconditioners proposed for Toeplitz problems were originally designed for Hermitian matrices generated by a smooth symbol f, for which estimates of the eigenvalues of the preconditioned matrix can be obtained and related to the convergence rate of the Krylov method [191,200]. More recently these sorts of results have been extended to a wider class of problems, known as (block) Generalised Locally Toeplitz (GLT) matrices [201,202]. For non-Hermitian matrices it is often possible to prove that the singular values and/or eigenvalues of the preconditioned matrix are clustered. Unfortunately, with the exception of LSQR or LSMR (or similar symmetrisations), it is not possible to link these results to convergence rates of Krylov subspace methods. An exception can be made in specific cases, e.g., [177], in which the condition number of the eigenvector matrix is also bounded. Another possibility is to use the specific Toeplitz structure to symmetrise, as described in Section 5.

3.4 | Time-dependent problems

Time-dependent problems present new, often significant, challenges for efficient numerical methods. The most common way to solve time-dependent PDEs is by a method-of-lines approach, which gives rise to a sequence of problems (one for each time step), that must be solved sequentially. In this case, preconditioners are typically based on those for the analogous steady problem. These time-dependent systems can sometimes be easier to solve than their steady counterparts, in the sense that matrix blocks can have more convenient spectral properties. This is because time-stepping typically introduces a positive definite matrix, e.g., a mass matrix or identity matrix scaled by the inverse of the time step. For large time steps, however, the effect is usually small, and a good preconditioner should work for the range of time steps appropriate for the given problem.

An alternative approach is to solve for all time steps simultaneously. This is sometimes called an "all-at-once" or "one-shot" method. Any parallel time integration scheme that can be described by a linear operator may be used as a preconditioner, with an overview of possible methods, e.g., domain decomposition and multigrid approaches, given in the survey by Gander [203]. Other techniques include matrix equation-based preconditioners [204] (with a flexible Krylov method if the matrix-equation solver is nonlinear) or, in the case of uniform time steps, preconditioners that exploit block Toeplitz structure [205,206]. A disadvantage of the all-at-once approach is that the sizes of all time steps need to be determined in advance. However, this can still be useful in certain settings, e.g., predictor–corrector methods [207] or in optimisation problems [208–210].

3.5 | Problems with heterogeneous coefficients

Problems with heterogeneous coefficients are in general more difficult to precondition than those without. They occur frequently when modelling real-world phenomena, often arising in nonlinear problems. The simplest such problem is a diffusion problem

with varying diffusivity, i.e., (3) with nonconstant α. An elementary option that can be effective for such problems is to first scale the coefficient matrix by its diagonal[211], which can reduce the effect of α, and then apply a standard Poisson preconditioner. Alternatively, many of the techniques described above, such as multigrid[212,213] and domain decomposition[88,211,214], have been adapted to deal with problems with heterogeneous coefficients by, e.g., using more sophisticated smoothers, scalings, or coarse grid solves. Recently, insights into the performance of simple Laplace preconditioners for (3) were obtained via careful eigenvalue analysis[67]; this, similarly to[211], showed that in the case of heterogeneous problems, looking at the condition number of the preconditioned coefficient matrix alone may not be indicative of the effectiveness of the preconditioner. Finally, we note that it may be advantageous to reformulate the system. For example, (3) can be rewritten as a system of first-order equations, for which preconditioners may be more readily constructed[168,215].

3.6 | Other methods

A *sparse approximate inverse (SPAI) preconditioner* for A may be defined as $P^{-1} = R$, where R minimises $\|I - AR\|$ subject to R also possessing some sparsity pattern. This is most commonly considered in the Frobenius norm (as initially investigated in[216], see also[217–223]), for which the minimisation problem then reduces to separate least squares problems for each column of R that may be solved in parallel. Among many important works on the subject, we refer to[4, Chapter 5] for a detailed summary of such preconditioners, to[224] for a description of Grote and Huckle's successful SPAI algorithm for dynamically defining a sparsity pattern for R, to[225] for a discussion of nonsymmetric linear systems, and to[226–228] for comparative studies of different SPAI preconditioners and sparsity structures. In the context of PDEs, a key application is the use of Frobenius norm-based preconditioners for large dense systems resulting from boundary integral formulations of electromagnetic problems[217,218,229,230]. Other important applications of approximate inverse preconditioners arise in elasticity problems[219,222,223], magnetohydrodynamics[231], and acoustic simulations[220].

It is also possible to construct preconditioners based on fixed point iterations. Although simple approaches based on Jacobi and SOR are not often used on their own for PDE problems, splittings that are tailored to the PDE in question can form the basis of effective preconditioners. One example is a splitting according to the spatial dimension, which is the basis for the alternating directions implicit (ADI) method[232] and related approaches like the dimension splitting method[152,153] , and the Hermitian–skew-Hermitian splitting method (see Section 3.1.3).

A *deflation preconditioner* is a projection that aims to (approximately) remove a subspace responsible for slow convergence[19,86,233,234]. A common motivation is to remove the eigenspace associated with small eigenvalues, although in theory any subspace could be removed. On the other hand, it is also possible to precondition to augment the Krylov subspaces in which a solution is sought by adding search directions that accelerate convergence[19,234]. Closely related to these ideas are polynomial preconditioners, i.e., preconditioners of the form $P^{-1} = p(A)$, for some polynomial p. These aim to incorporate spectral information, so that the properties of $P^{-1}A\boldsymbol{x} = P^{-1}\boldsymbol{b}$, where $P^{-1}A = AP^{-1} = Ap(A)$, are more favourable[16, Section 12.3].

We have already seen how dense Toeplitz problems may be solved by Krylov subspace methods. Iterative methods can also be applied to other dense, but data-sparse, matrices stored in a format that enables fast matrix–vector products, where a matrix is data-sparse if it can be stored in some representation using far fewer than n^2 degrees of freedom. For PDEs, the most common structure is the presence of low-rank off-diagonal blocks. Intuitively, this is because far-range interactions tend to be relatively weak, and so can be well approximated by low-rank structures. Data-sparse formats include $\mathcal{H}$-matrices[136,235,236], $\mathcal{H}^2$ matrices[237], hierarchically semi-separable matrices[238], the fast multipole method[239], and recursive skeletonisation[240]. Often, by storing matrices in these formats, it is possible to accelerate matrix–vector products within Krylov subspace methods. However, it may also be possible to use these approaches for preconditioning[241,242].

We conclude this section by commenting on a technique that can be applied when a problem has two (or more) preconditioners that accelerate the convergence rate of a linear system. Whilst it may be difficult to formulate a new preconditioner that captures the good behaviour of each individual preconditioner, they can be combined within a *multipreconditioned iterative solver*[243,244]; this has been successfully used in the solution of parameter-dependent problems[245] and two-phase flow[246]. More generally, when dealing with sequences of problems it can make sense to reuse information from previously-computed systems, including preconditioners[19, Section 14] (see also Section 4.4).

4 | PRECONDITIONERS FOR OPTIMISATION PROBLEMS

Another major field of research which necessitates the solution of linear systems of equations describing a physical process is that of optimisation. As an illustration, if one wishes to solve the constrained *quadratic programming* problem:

$$\begin{aligned} \min_{\boldsymbol{x}} \ & \frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} \\ \text{s.t.} \ & B\boldsymbol{x} = \boldsymbol{c}, \end{aligned}$$

where A is symmetric positive semidefinite, one may construct the Lagrangian:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{2}\boldsymbol{x}^T H\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} + \boldsymbol{y}^T(B\boldsymbol{x} - \boldsymbol{c}).$$

Finding a stationary point of $\mathcal{L}$ then leads to a linear system exactly of the saddle point form (6), with $A = H$. Further matrices may arise within this system: for example, when solving problems with additional inequality constraints imposed on $\boldsymbol{x}$ using an *interior point method*, terms arising from a barrier function enforcing these constraints lead to an additional diagonal matrix which is added to A[247]. Further matrix terms can also arise in the $(2,2)$-block of the saddle point system as a result of regularisation terms applied to solve optimisation problems.

Often, for example in the case of linear programming problems[248–251], A can be a diagonal matrix as $H = 0$. In this case it is helpful to reduce the system to the equivalent Schur complement system (8) (often referred to as the normal equations within the optimisation literature), provided A is invertible.

In optimisation it is important to consider the advantages and disadvantages of solving the Schur complement system or the saddle point system, especially if A^{-1} has a particularly convenient structure. The Schur complement is guaranteed to be symmetric positive semidefinite; however if B has full row rank the reduced system may be significantly more ill-conditioned than the matrices A and B themselves, and moreover with increased density. Examining instead the saddle point system circumvents these issues, however this system is indefinite and is (often significantly) larger than the Schur complement matrix.

Another important consideration is that, in optimisation, frequently very little is known about the structure of the linear system, apart from the matrix being sparse in many applications (which in general we assume in this section). This contrasts with preconditioning PDE problems, where some more descriptive properties of the system are generally known *a priori*, due to the features of the analogous infinite-dimensional problems for instance. Therefore, the strategies used to precondition systems in optimisation often need to be more general, to account for varying structures arising in the sub-blocks of the linear systems.

In this section we give a representative summary of some classes of methods used to precondition linear systems arising from optimisation problems in the literature. We wish to emphasise that many strategies arising from the two areas of PDEs and optimisation can have considerable similarities and can sometimes be transferred between the fields, notwithstanding the particular challenges that optimisation problems pose. For instance, by exploiting the saddle point structures of matrices that often result, block preconditioners for PDEs like those in Section 3.2 can also be applied to linear systems arising from constrained optimisation problems. It is also possible to apply many of the approaches outlined in this section to systems arising from PDEs, and we give examples in Section 4.6, which considers optimisation problems with PDEs themselves acting as constraints. We present the following methods for optimisation problems in this way because we wish to provide a self-contained summary of some preconditioners within the field of optimisation, which we believe cannot be found in much of the literature, and which highlights in particular some methods that require relatively little knowledge about the precise structures of the matrices being tackled. In particular, some of the preconditioners we propose—notably incomplete factorisations—can be applied to *any* coefficient matrix A, not just the optimisation problems considered in this section.

4.1 | Incomplete factorisations

As discussed previously, solving large-scale problems using direct methods can rapidly become impractical as the dimension of the system increases. Computing an LU factorisation of a matrix $M = LU$ (or $M = LDU$) can result in L, U being significantly less sparse than M, a feature known as *fill-in*, although there are strategies in place to mitigate this including reorderings[1,252,253]. The same also applies to computing a Cholesky factorisation $M = LL^T$ of a Hermitian positive definite matrix (or a factorisation $M = LDL^T$ of systems that need not be positive definite). We note that M could denote a matrix relating to the entire system being solved, or a sub-block of it.

The idea of *incomplete factorisation* is that one attempts to form an approximation of such a factorisation, that at the same time is cheap to apply. As with preconditioning in general, this necessitates a trade-off between computational efficiency and accuracy

of approximation, which in this case corresponds to how aggressively one may drop fill-in versus how potent an approximation results from this strategy. Essentially, algorithms are required that determine relevant criteria for the dropping of fill-in, based on position in the matrix, or value/magnitude of the entry. An incomplete Cholesky factorisation of a matrix M then leads to

$$M = LL^T + E,$$

where E is the error induced by the dropped terms. The same applies to incomplete LU (ILU) factorisation. There has been a substantial amount of research into devising incomplete LU and incomplete Cholesky factorisations, including within preconditioners [252,254–258].

One possible choice is to select sparsity patterns for the incomplete Cholesky or ILU factorisations which are the same as the matrix M itself. This leads to the ILU(0) or IC(0) factorisations. While these will be comparatively cheap to work with, they may contain rather little information about the matrix M as a result. However, this has nonetheless been shown to be effective for preconditioning certain classes of matrices [13,259]. This methodology can be generalised to IC(ℓ) or ILU(ℓ) preconditioners (where ℓ is a positive integer) to achieve different levels of fill-in. We refer to [4, Sections 3 & 4] for a thorough survey of such methods, as well as research on the existence and stability of block factorisations and preconditioners.

We highlight that triangular factorisations need not be computed using Cholesky for positive definite matrices: for example in [260] Benzi and Tůma compute such a factorisation by orthogonalisation with respect to the inner product generated by M. Among much other research in the area of incomplete factorisation preconditioners, there has also been a focus on multilevel and multifrontal preconditioners [257,258,261,262], some of which can give rise to parallel computing technologies.

Within optimisation, incomplete factorisations form a valuable tool, which can be used as an effective preconditioner for the normal equations, or as an approximation for one block of a saddle point system. There have been many such applications within the field, for example preconditioning linear systems arising from the solution of linear programming problems [248,263] (see also [249]), nonlinear optimisation [264,265], least squares problems (as in Section 4.5), and many others. Incomplete factorisations are also widely used in applications beyond optimisation, since they can be computed using only matrix entries.

4.2 | Constraint preconditioning

One key class of preconditioners for saddle point systems arising in optimisation, of the form (6), is that of *constraint preconditioners* [266–268]:

$$\mathcal{P} = \begin{bmatrix} \widehat{A} & B^T \\ B & 0 \end{bmatrix}. \tag{10}$$

These preconditioners are so called because at each iteration of an appropriate Krylov method, the constraints (corresponding to the matrix B) are exactly satisfied in exact arithmetic [266,269]. If A is symmetric positive definite on the kernel of B, which for this discussion we assume is of full row rank, then $\widehat{A}$ is typically selected to be another symmetric positive definite matrix.

We highlight that $\mathcal{P}$ is not a symmetric positive definite matrix, and yet it may be applied within Krylov methods such as CG and MINRES: this being equivalent to the *projected CG* and *projected MINRES* algorithms [269], as the preconditioner is positive definite on the constraint manifold [266]. Furthermore, all eigenvalues of the preconditioned linear system $\mathcal{P}^{-1}\mathcal{A}$ that are not equal to one, are equal to those of $\widehat{A}^{-1}A$ [267]. An effective constraint preconditioner requires a method to construct a matrix of the form (10), ideally such that the resulting matrix $\widehat{A}$ is a potent approximation of A and such that its inverse is cheap to apply to a vector.

One widely used approach for constructing such a preconditioner is to make use of a *Schilders factorisation* [9,270,271], whereby one decomposes (possibly after reordering of the linear system) $B = [B_1 \ B_2]$ such that B_1 is invertible, then takes as a preconditioner:

$$\mathcal{P} = \begin{bmatrix} B_1^T & 0 & L_1 \\ B_2^T & L_2 & G \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} D_1 & 0 & I \\ 0 & D_2 & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} B_1 & B_2 & 0 \\ 0 & L_2^T & 0 \\ L_1^T & G^T & I \end{bmatrix}.$$

This factorisation is guaranteed to equate to a matrix of the form (10), for any D_1, L_1, G of the correct dimensions, and for nonsingular matrices D_2, L_2. The key task at this point is to construct matrices that well approximate the $(1,1)$-block of the saddle point system (6). We note that alternative factorisations may also be derived, for instance based on null-space factorisations (see [272] for a summary of some such approaches, and [151] for a statement in the context of constraint preconditioning).

Constraint preconditioners may also be constructed for generalised saddle point systems, of the form (6) but with a non-zero $(2,2)$-block. Such systems can arise from optimisation problems with additional regularisation, and associated constraint preconditioners are considered in, e.g., [265,270,271,273].

Within the field of optimisation, constraint preconditioners have been devised for linear and quadratic programming problems in [247,248,266,267,273–275], and additionally for nonlinear optimisation problems in [265,267,268,271,276,277].

We highlight a number of further studies into the field of constraint preconditioning. Theoretical properties and convergence results are studied in detail in [278], constraint preconditioners for nonsymmetric indefinite linear systems are analysed in [279], and a new limited memory approach with the objective of improving the stability and robustness of constraint preconditioning is described in [280]. Such preconditioners have also been devised for optimisation problems with PDE constraints [281], including within the field of topology optimisation involving PDEs [282].

4.3 | Augmented Lagrangian preconditioning

Another class of preconditioners for saddle point systems (6) from optimisation are so-called *augmented Lagrangian preconditioners*:

$$\mathcal{P} = \begin{bmatrix} A + B^T W^{-1} B & 0 \\ 0 & W \end{bmatrix}. \tag{11}$$

This type of preconditioner is particularly useful if the $(1,1)$-block A of (6) is singular, but is positive definite on the kernel of B. In this case many classical saddle point preconditioners cannot be straightforwardly applied, but (11) is invertible for any symmetric positive definite matrix W, and furthermore is symmetric positive definite so may be applied within the MINRES algorithm. The eigenvalues of the resulting preconditioned system are analysed in [283], demonstrating many enjoyable properties, and much original work on such preconditioners was tailored to solving PDEs, see [143,284] for instance. Of course, the effectiveness of the preconditioner depends on a suitable choice of W. One frequent choice is that of $W = \omega^{-1} I$, which is analysed in detail in [285], especially in relation to the selection of the parameter $\omega > 0$.

One link with the field of optimisation (relating in particular to the naming of the preconditioner) is through the augmented Lagrangian idea in optimisation [9,286,287], particularly useful in cases where A is an ill-conditioned matrix, which notes that the system (6) is equivalent to the system:

$$\begin{bmatrix} A + B^T W^{-1} B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} + B^T W^{-1} \boldsymbol{c} \\ \boldsymbol{c} \end{bmatrix}.$$

Indeed the augmented Lagrangian preconditioner has been applied widely in optimisation: one of the first such applications was to primal–dual interior point methods for linear and quadratic programming problems [288], see also [285,289] for other applications in optimisation. The augmented Lagrangian preconditioner idea has been extended to nonsymmetric saddle point systems in [143,290], and has also found applicability to regularised systems where the $(2,2)$-block of the saddle point system is non-zero [289,291]. It is also possible to construct block triangular analogues of (11) for saddle point systems [9,143,288,290,291].

4.4 | Low-rank updates of preconditioners for sequences of linear systems

A research area of active interest of late within the optimisation community is the solution of sequences of linear systems

$$M^{(k)} \boldsymbol{x} = \boldsymbol{b}^{(k)}, \qquad k = 0, 1, 2, \dots,$$

each of which may be of saddle point form (6), for instance. These can arise, for example, through the solution of nonlinear equations or an optimisation problem using a Newton- or Broyden-type method, yielding a sequence of sub-problems of the form

$$J(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = -F(\boldsymbol{x}_k), \qquad k = 0, 1, 2, \dots,$$

where $J(\boldsymbol{x}_k)$ is the Jacobian evaluated at the current iterate $\boldsymbol{x}_k$. It has been found that constructing *low-rank updates* of preconditioners may be highly effective. For instance, assuming one starts with an effective preconditioner $\mathcal{P}_0$ for $J_0 := J(\boldsymbol{x}_0)$, one may successively 'correct' preconditioners at each iteration using rank-1 updates for $\mathcal{P}_k$:

$$\mathcal{P}_{k+1} = \mathcal{P}_k + \boldsymbol{u}\boldsymbol{v}^T, \qquad k = 0, 1, 2, \dots,$$

and then use the Sherman–Morrison formula to apply $\mathcal{P}_{k+1}^{-1}$ to $J_{k+1} := J(\boldsymbol{x}_{k+1})$ (see [292] for a discussion). This motivation may of course be generalised to updated preconditioners of higher ranks. In a similar spirit to that of SPAI preconditioners, for instance, such preconditioners may be constructed, with a reasonable objective being that $\|I - \mathcal{P}_k^{-1} J_k\|$ is bounded and small.

Crucial early work in this direction includes the papers of Martínez [293,294], where improving a preconditioner through low-rank updating is suggested. The preconditioners derived in these works were termed *secant preconditioners*, because they were

required to satisfy the "secant property":

$$\mathcal{P}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k) = F(\boldsymbol{x}_{k+1}) - F(\boldsymbol{x}_k).$$

The idea of updating a previously computed matrix to accelerate the CG method, by saving information in the form of a quasi-Newton matrix, is also examined in [295], based on earlier research in this area such as in [296,297].

Preconditioners for the inexact Newton method were devised, for example, in [292,298], based on Broyden-type rank-1 updates [292] and BFGS rank-2 updates [298], respectively. Preconditioning of nonsymmetric systems using updates of incomplete factorisations is discussed in [299], and updated preconditioners which may be applied in a matrix-free setting are derived in [300].

Another research area of interest in optimisation is the construction of low-rank updates of constraint preconditioners for quadratic programming problems [301–303]. In particular the paper [303] extends work in [304], where limited memory preconditioners for symmetric positive definite systems are considered. We also highlight the work [305], where low-rank updates of preconditioners are constructed for data assimilation, with application to the 4D-Var weather prediction model.

Finally, we note that low-rank updates of preconditioners can also be helpful when solving sequences of linear systems within applications other than optimisation, for example the solution of PDEs (particularly time- or parameter-dependent problems), nonlinear problems, eigensolvers, and rational Krylov methods for matrix functions. In the particular, but important, case that these systems are all of the form $A = M + \psi I$, with ψ changing, modifying a given preconditioner for such a sequence is considered in, e.g., [306–308]. We refer to [309] for a recent survey of low-rank updates of preconditioners, which considers a number of the above applications.

4.5 | Preconditioners for least-squares problems

Another specific class of problems that has gained much interest in recent years is that of *least-squares problems*:

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{b} - A\boldsymbol{x}\|_2 . \tag{12}$$

Differentiating the square of the functional in (12) with respect to $\boldsymbol{x}$ to find the stationary point leads to the *normal equations*:

$$A^T A\boldsymbol{x} = A^T \boldsymbol{b}, \tag{13}$$

which in turn may easily be shown to be equivalent to solving the *augmented system*:

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}. \tag{14}$$

Similarly to when we consider the respective merits of solving saddle point systems of general form (6) versus Schur complement systems (8), there are advantages and disadvantages of solving the augmented system and the normal equations.

We first refer to two important articles summarising developments in preconditioning for least-squares problems: [310] considers the solution of the normal equations using the conjugate gradient for least squares (CGLS) method before proposing new strategies, and [311] presents a thorough numerical study of preconditioners for the normal equations and the augmented system. We summarise the findings of these two papers very briefly here.

A number of highly effective strategies exist for preconditioning the normal equations (13). Apart from Jacobi (diagonal) preconditioning (see [311]), one may apply incomplete Cholesky methods (provided A has full column rank), including limited memory approaches [312–314]. One may also make use of incomplete orthogonalisation methods [315,316], including incomplete QR (IQR) decompositions [317,318] based on Gram–Schmidt orthogonalisation with no dropping in Q, a method that is termed *compressed incomplete modified Gram–Schmidt* (CIMGS). IQR methods have also been extended to multilevel IQR (MIQR) approaches [319]. Another valuable class of approaches for the normal equations are robust (RIF) and balanced (BIF) incomplete factorisations, see [260,310,320].

Incomplete factorisation strategies also exist for preconditioning the augmented system (14), such as signed incomplete Cholesky factorisations [321], or incomplete LDL^T factorisations [322].

We also highlight a number of other developments regarding the iterative solution of least-squares problems, including the limited memory preconditioner for the normal equations developed in [2], preconditioners based on approximate inverses [323,324], the solution of such problems based on augmented matrix methods [325, Chapter 7], and preconditioners based on LU factorisations for both the normal equations [326] and the augmented system [327]. Specialised preconditioners can also be devised for least-squares problems with more specific properties, such as weighted Toeplitz least-squares problems [328] (cf. Section 3.3).

4.6 | Link to PDEs and PDE-constrained optimisation

As above, although PDEs and optimisation form two separate branches of mathematics, there is considerable scope for many of the preconditioning strategies presented in this section to be used to tackle the linear systems discussed in Section 3 that arise from discretisations of PDEs. For example, ILU factorisations have been extensively applied to PDE problems (see [329,330] for instance), and incomplete factorisations find considerable applicability as smoothers within multigrid [68]. Constraint preconditioners [278,331–333] and augmented Lagrangian preconditioners [143,144,284,334] have also been widely used when solving PDEs.

However, it can also be that the nature of PDE and optimisation problems are distinct, and so one must carefully tailor preconditioned iterative solvers to the problem at hand. For example, PDE problems typically offer more specific matrix structures which may be exploited by preconditioners, whereas optimisation solvers frequently need to be more general. The differing philosophies between these two classes of problems is one reason why the methods presented in Sections 3 and 4 are distinct.

We conclude this section with one particular class of problems in which PDEs and optimisation problems intersect: that of *PDE-constrained optimisation*. To give an illustration of iterative solvers for the resulting linear systems, consider the *Poisson control* problem:

$$\begin{aligned} \min_{u,f} \ & \frac{1}{2}\|u-\widehat{u}\|^2_{L^2(\Omega)} + \frac{\beta}{2}\|f\|^2_{L^2(\Omega)} \\ \text{s.t.} \ & -\nabla^2 u = f \ \text{ in } \ \Omega, \end{aligned} \tag{15}$$

given a domain $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, a function $\widehat{u}$, and a positive regularisation parameter β, as well as appropriate boundary conditions. The variables u and f are often referred to as the *state variable* and *control variable*, respectively. Discretising this problem using finite elements leads to a system of the form [52,335]:

$$\left[\begin{array}{cc:c} Q & 0 & K \\ 0 & \beta Q & -Q \\ \hdashline K & -Q & 0 \end{array}\right]\left[\begin{array}{c} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \hdashline \boldsymbol{y} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \hdashline \boldsymbol{c} \end{array}\right],$$

where Q denotes a mass matrix and K a stiffness matrix, and with $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ corresponding to the discretisations of u, f.

Poisson control problems of this type have been considered from both the infinite-dimensional [52,55,167,174,336] and finite-dimensional [281,335,337,338] points of view. In particular by viewing the problem on the infinite-dimensional level, for instance by considering nonstandard norms, and then discretising the resulting infinite-dimensional interpretation of the preconditioner, one may arrive at the optimal preconditioner [52,55]:

$$\mathcal{P} = \begin{bmatrix} \beta^{1/2}K + Q & 0 & 0 \\ 0 & \beta Q & 0 \\ 0 & 0 & \beta^{-1}(\beta^{1/2}K + Q) \end{bmatrix}.$$

This methodology may also be applied in the setting of constraint preconditioners [336], for instance. On the other hand, by considering the properties of the saddle point system purely on the discrete level, one may also apply the preconditioner [335]:

$$\mathcal{P} = \begin{bmatrix} Q & 0 & 0 \\ 0 & \beta Q & 0 \\ 0 & 0 & (K+\beta^{-1/2}Q)Q^{-1}(K+\beta^{-1/2}Q) \end{bmatrix},$$

which is also optimal. This approach may itself be extended to block triangular preconditioners [335,338], for instance. Both preconditioners lead to provably mesh- and β-independent solvers for the problem (15), and we note that both preconditioning 'routes' lead to different (if related) structures.

Interestingly, both the 'infinite-dimensional' and the 'discrete' routes are known to deliver mesh- and parameter-robust solvers for particular PDE-constrained optimisation problems, for which the other approach as yet cannot. For instance, deriving preconditioners on the infinite-dimensional level and then discretising these operators has led to potent solvers for the Stokes control problem [52], and a number of time-periodic problems [339]. On the other hand, considering the optimisation problem using a purely discrete (saddle point) framework has similarly generated robust solvers for convection–diffusion control problems [340], and distributed time-dependent problems [341].

We highlight that there also exist a range of other methods for the Poisson control problem (15), e.g., modified Hermitian and skew-Hermitian (MHSS) methods [342], nonsymmetric preconditioners which are not of block triangular form [343], and constraint preconditioners [281]. Of course adjustments can be made to problems of the form (15), that lead to more complex structures.

Preconditioning linear systems arising from such problems is an active research area, with too many references to mention here. To give an illustration, there has been work on deriving preconditioners for problems with additional algebraic box constraints on the state and/or control variables [344–348], problems with limited observation [166], and parameter estimation problems [349], along with linear systems arising from a range of other scientific applications.

5 | PRECONDITIONERS WITH "NONSTANDARD" GOALS

In the previous two sections we have summarised a range of problems for which preconditioned iterative methods have found considerable utility, and a number of solution strategies that have been employed. In each case the key objective was to achieve rapid and robust convergence of an iterative method for solving a linear system $A\boldsymbol{x} = \boldsymbol{b}$, in a way which cannot be guaranteed by a direct method or an unpreconditioned iterative scheme. However this is not the limit of the capability of preconditioners. For instance one may also solve problems of a different general form using preconditioned solvers, such as eigenvalue problems, or alter the structure of an original problem statement. One may also have an additional motivation behind implementing a preconditioner in the first place: for instance reducing computer storage requirements, quantifying uncertainty in the solution, or implementing a method on high performance computing (HPC) architectures. In this section we very briefly survey valuable research which has been undertaken on preconditioning strategies for achieving such "nonstandard" objectives.

▷ **Eigenvalue problems:** Consider the problem of finding a simple interior eigenvalue λ of a symmetric matrix A, and its associated eigenvector $\boldsymbol{x}$. Algorithms for this include inverse iteration or Rayleigh quotient iteration (RQI), both of which require the solution of linear systems

$$(A - \sigma I)\boldsymbol{y}_k = \boldsymbol{x}_k, \tag{16}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{x}_{k+1} = \boldsymbol{y}_k/\|\boldsymbol{y}_k\|$ are successive approximations to an eigenvector $\boldsymbol{x}$.

When A is large, and matrix–vector products with A are easy to apply, it is attractive to solve (16) using a Krylov subspace method, such as MINRES. This leads to an inner–outer scheme, in which MINRES is the inner method, and inverse iteration or RQI forms the outer iteration. Although it is still possible to achieve the outer iteration convergence rate obtained when (16) is solved exactly, this generally necessitates that a more stringent MINRES tolerance is used as the outer iteration proceeds [350–354]. For a generic linear system, decreasing the tolerance would increase the number of MINRES iterations, but perhaps surprisingly this is not the case here: as the outer iteration proceeds the right-hand side in (16) better approximates an eigenvector of $A - \sigma I$, making the system easier to solve. As a result, the number of inner iterations remains bounded [350,355]. (This clearly illustrates the role the right-hand side plays in determining the convergence rate of a Krylov subspace method.)

This property is usually destroyed when preconditioners are applied: although a good preconditioner will be effective at early outer iterations, decreasing the preconditioned MINRES tolerance typically increases the number of inner iterations. This is simply due to the fact that the preconditioned right-hand side no longer approximates an eigenvector of the preconditioned system. To remedy this, it is necessary to choose the preconditioner so that the preconditioned right-hand side approximates an eigenvector. This is a rather unorthodox view of preconditioning, but here proves to be just what is needed.

To see how this is achieved, consider the idealised situation that $(A - \sigma I)\boldsymbol{y}_k = \boldsymbol{x}$, i.e., that the right-hand side is precisely the desired eigenvector. Then, for any positive definite preconditioner P, if $\boldsymbol{u} = (A - P)\boldsymbol{x}$,

$$P_{\text{tuned}} = P + \frac{\boldsymbol{u}\boldsymbol{u}^T}{\boldsymbol{x}^T\boldsymbol{u}}$$

is such that $P_{\text{tuned}}\boldsymbol{x} = A\boldsymbol{x}$. It can then be shown that preconditioned MINRES applied to this system would converge in exactly one step [355]. Although such a preconditioner is infeasible in general, replacing $\boldsymbol{x}$ by $\boldsymbol{x}_k$ and $\boldsymbol{u}$ by $\boldsymbol{u}_k := (A - P)\boldsymbol{x}_k$ results in a preconditioner for which the number of inner iterations remains bounded as the outer iterations proceed. Moreover, by making use of the Sherman–Morrison formula, applying P_{tuned} is not significantly more challenging than applying P [180, Section 2.1.4]. This idea has been analysed for nonsymmetric generalised eigenvalue problems [356], and applied in inverse subspace iteration [357] and Arnoldi-type eigensolvers [358,359].

We have focussed here on preconditioning the inner solves within iterative eigenvalue solvers. However, for some Krylov-based eigenvalue solvers, e.g., certain Lanczos-based methods, it is possible to precondition the eigenvalue solver itself to accelerate convergence to the desired eigenvalues [360–362].

▷ **Symmetrisation:** As we saw in Section 2, MINRES is only applicable when the coefficient matrix is symmetric, while CG additionally requires positive definiteness. It is somewhat surprising, therefore, that at least in theory any linear system with

a real nonsymmetric coefficient matrix $A \in \mathbb{R}^{n\times n}$, can be solved using MINRES (or possibly CG) after appropriate preconditioning. This is because A can always be factored as $A = Y^{-1}T$ with Y, T symmetric[363], which implies that YA must be symmetric. Here, the preconditioner Y serves not to improve the convergence rate but to enable MINRES or CG to be used, with their short-term recurrences, optimality properties, and comprehensive convergence theory. All is not rosy, unfortunately, since computing Y usually requires the Jordan normal form of A, which is prohibitively expensive to compute, and almost always results in a dense matrix Y that is too costly to apply.

There are some exceptions, however: for certain structured matrices, a sparse Y is obtainable without recourse to the Jordan form. In these cases symmetrisation is an attractive option. For example, if A is Toeplitz and

$$Y = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix},$$

then YA is a Hankel, hence symmetric, matrix. This well-known fact has been exploited to develop solvers that are typically faster than GMRES applied to the original system, thanks to the short-term recurrences of MINRES[364–366]. We stress that Y does not, in general, improve the convergence rate[367,368]; a second, symmetric positive definite, preconditioner is almost always required. However, in contrast to the nonsymmetric system, the preconditioner choice can be guided by MINRES convergence theory, and mathematical guarantees of fast convergence can be obtained. More generally, any persymmetric matrix is symmetrised by the reverse identity matrix, while $2n \times 2n$ Hamiltonian matrices are symmetrised by $\left[\begin{smallmatrix} 0 & I_n \\ -I_n & 0 \end{smallmatrix}\right]$.

A matrix that is symmetrised by Y as described above is self-adjoint with respect to the symmetric bilinear form[369, Chapter 1] $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_Y = \boldsymbol{y}^T Y \boldsymbol{x}$. When this bilinear form is an inner product it can be used to define Krylov subspace methods with optimality properties, such as Y-CG or Y-MINRES[19, Section 13]. Even for standard CG and MINRES the choice of preconditioner essentially fixes the inner product in which the method is defined[370]. By utilising a different inner product, it may be possible to find a preconditioner P such that $P^{-1}A$ is self-adjoint in the Y-inner product, even if A and/or P are not symmetric[371,372].

Combinations of preconditioner and inner product need not only result in symmetry to be advantageous. For example, when the symmetric part of A, namely $Y = (A + A^T)/2$, is positive definite, $Y^{-1}A$ is the sum of the identity matrix and a matrix that is skew-adjoint with respect to the Y-inner product. In this case a short-term recurrence method can be applied[373,374].

▷ **Ill-posed problems:** For ill-posed problems $A\boldsymbol{x} = \boldsymbol{b}$, the matrix A typically has singular values that decay to zero and a right-hand side that is affected by noise, i.e., $\boldsymbol{b} = A\boldsymbol{x}_{\text{true}} + \boldsymbol{\eta}$. For such problems, we would like to approximate $\boldsymbol{x}_{\text{true}}$. However, the ill-conditioning of A means that approximations to $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{true}}$ may differ greatly even when $\|\boldsymbol{\eta}\|$ is small. Under certain conditions, iterative methods applied to ill-posed problems *do* approximate $\boldsymbol{x}_{\text{true}}$ at early stages, but after this noise dominates, at which point approximations to $\boldsymbol{x}_{\text{true}}$ get worse again. When this phenomenon, which is known as semi-convergence[375–377], occurs the iterative method itself acts a regulariser of the problem.

A good preconditioner P will also make it easier to approximate $\boldsymbol{x}_{\text{true}}$, but this does not mean that we should choose $P \approx A$. Indeed, if P is a good representation of A it will almost certainly be ill-conditioned, and so its application may significantly amplify errors at early iterations. Instead, P should also regularise the problem somehow. Typically, one wants to damp the high-frequency components associated with small eigenvalues or singular values of A. A common option is to set these high-frequency components to one[378–380], but another option is to remove them completely via a singular preconditioner[381,382] or a projection[383]. We note that other types of regularisation, e.g., Tikhonov, can also be enforced via preconditioning[384,385]. A different interpretation is to develop preconditioners for ill-posed problems using a Bayesian framework[386] (see the discussion on probabilistic solvers below).

▷ **Randomised preconditioning:** An area of significant recent interest in the linear algebra community is that of randomised linear algebra. In particular, randomised preconditioners may be developed for least-squares problems of the form (12). A key work is that of[387], where overdetermined problems are considered (i.e., where $A \in \mathbb{C}^{m\times n}$, $m \geq n$), although the authors' methods can be extended to underdetermined regression. Here the problem (12) is replaced by a problem of minimising $\|AP^{-1}\boldsymbol{y} - \boldsymbol{b}\|$ over vectors $\boldsymbol{y}$, whereupon the solution $\boldsymbol{x} = P^{-1}\boldsymbol{y}$. The solver incorporates a randomised transformation, with the preconditioner based on a permutation of the R factor from the QR decomposition of the transformed matrix A. Another important work is[388], where the randomised methods may be viewed as preconditioning the matrix A and vector $\boldsymbol{b}$ with a data-independent random matrix P. In[389], a high-precision preconditioned solver is presented, where the rows and columns of A are blended, at which point a random sampling of rows is taken. We also refer to least squares solvers based on random normal

projection in [390,391], with [391] deriving a parallel solver for strongly over- or underdetermined systems, and randomised preconditioners for overdetermined ℓ_p regression problems (which involve minimising $\|\boldsymbol{b} - A\boldsymbol{x}\|_p$ for $1 \leq p \leq \infty$) applied within Stochastic Gradient Descent-like solvers in [392].

▷ **Probabilistic linear solvers:** A further concept of considerable recent interest has been a Bayesian interpretation of preconditioning. A key early work in this direction is [393], where preconditioners for ill-posed systems of the form $A\boldsymbol{x} = \boldsymbol{b}$ (where A is of ill-defined rank) are obtained from the covariance matrix of the solution interpreted as a random variable. These were termed *priorconditioners*. Building on this, a statistical interpretation of left- and right-preconditioners was provided in [386], specifically connecting such preconditioners to covariance matrices of noise and prior, respectively, in certain settings. A priorconditioned CGLS method is applied to inverse problems from electrical impedance tomography in [394].

We also highlight research in [395,396] which provides novel derivations of CG methods, outputting probability measures which may be used to quantify uncertainty in the solution. In particular [395] extends probabilistic interpretations of secant methods and mentions the potential for preconditioning the derived method, and [396] discusses in detail preconditioners for the authors' *BayesCG* method, which are thought of as priors. As outlined in [397], the conceptual difference is that in [395] a posterior is constructed on A^{-1} whereas in [396] a posterior is constructed on the solution $\boldsymbol{x}$. The work [397] unifies the two approaches presented in [395] and [396], provides an interpretation of left- and right-preconditioning, and extends the idea to the GMRES algorithm.

▷ **Preconditioning and tensor methods:** An important consideration when solving linear systems, for instance those of high-dimensional PDEs, is that of data storage on a computer, which can become prohibitive unless specialised schemes are devised in order to combat this. One area of interest is that of utilising low-rank tensor decompositions alongside preconditioned solvers, for linear systems which possess some underlying tensor-product structure. Such techniques assume in general that the input data and solution vector are approximable within such a low-rank tensor format, for instance the *hierarchical Tucker format* (see [398,399]), the *Tensor Train (TT) format* (see [400]), and its extension of the *Quantized Tensor Train (QTT) format* (see [401,402]). Additionally, for linear systems with Kronecker product structure, e.g., arising from certain high-dimensional PDEs, it is also possible to use Krylov subspace methods designed for tensors or matrix equations [403,404]. Among many valuable works on tensor methods for the numerical solution of differential equations, coupled with preconditioners, we refer to [405–408] (see also [403] for the construction of Krylov subspace methods). Here, low-rank variants of iterative schemes are derived, where the iterates are replaced by low-rank tensors, and the potency of these schemes depends on the derivation of effective preconditioners. Additionally, recent research has been undertaken on tensor-based schemes for PDE-constrained optimisation problems [409,410], as well as FDE-constrained optimisation problems [411,412].

▷ **High-performance computing:** Some problems are so large and complex that they necessitate high performance computing and heterogeneous architectures. This brings new challenges for iterative methods: memory is usually limited and communication, particularly between nodes and processors, is slow relative to the speed of floating point operations. Since sparse matrix–vector products and vector operations are typically communication-bound, i.e., their speed is limited by the speed of moving data, it is not straightforward to improve the efficiency of Krylov methods in HPC environments [413–415]. On the other hand, since Krylov methods rely on a few simple operations, there are opportunities to utilise, e.g., graphics processing units (GPUs) to improve efficiency [416].

Although preconditioned iterative methods for HPC applications should also aim to solve the linear system as quickly as possible, rather than reducing the number of floating point operations the aim is typically to exploit parallelism and reduce communication costs. These goals can be achieved by improving data locality, thereby enabling more operations to be performed on data before communication is required, and/or by asynchronous communication and careful task scheduling, e.g., by using directed acyclic graphs to map data dependencies [417]. Unfortunately, some of the best preconditioners for serial computations, e.g., multigrid, are naturally communication bound as a result of coarse grid computations and grid transfer operations. Consequently, much work has gone into making these methods more suitable for HPC [418,419]. On the other hand, other methods including domain decomposition [89,90,92–95] and sparse approximate inverse preconditioners [420,421] have long been associated with high performance computing because they traditionally require relatively little communication and are highly parallelisable. However, perhaps counterintuitively, newer versions of these algorithms may increase communication and/or reduce parallelism to improve overall efficiency on cutting-edge computer architectures. For example, domain decomposition methods may also include a multilevel component that induces global coupling [422], while SPAI preconditioners may, e.g., increase sequential computation to improve efficiency [423]. Additionally, it may be advantageous to tailor kernels to GPUs (see, e.g., [424]).

Though the construction and application of incomplete factorisation preconditioners can be problematic on parallel architectures, performance can be improved by, e.g., using an asynchronous iterative method to form the factors [425] and exploiting

techniques from sparse approximate inverses [426]. Other preconditioners have been developed with extreme-scale computing in mind; these may require more computation than alternatives, and hence are slower for serial or moderately parallel computations, but are better able to utilise many-core architectures [242,427].

6 | CONCLUSIONS

In this article we have given a short overview of the multitude of applications of preconditioning within scientific fields. The key goal of preconditioned iterative solvers is to ensure the rapid and robust solution of linear systems of equations, particularly in cases where this cannot be achieved by a direct method or unpreconditioned iterative solver, however in a number of cases one may have additional objectives in mind, as in Section 5. Research has ranged from the design of preconditioners for matrices about which a number of specific properties are known, particularly in PDE applications (as in Section 3), or more general black-box solvers, especially in the study of optimisation problems (as in Section 4). Of course, due to the aim of providing a brief, overarching view on the vast field of preconditioning, we have omitted discussions on a number of other application areas, for example network problems and new developments in data science.

There does not yet exist a 'universal' preconditioner, indeed well-constructed preconditioners tend to capture the properties arising from the particular application being considered, sometimes including the physics of the process itself. By employing this philosophy, many researchers have made significant contributions to the study of scientific processes through tackling the resulting linear systems.

ACKNOWLEDGMENTS

The authors would like to thank three anonymous referees for their valuable comments and helpful suggestions. The authors would also like to express their gratitude to Luca Bergamaschi, Victorita Dolean, Jacek Gondzio, Zdeněk Strakoš, Jemima Tabeart, Ioana Vaduva, and Andy Wathen for their useful feedback on a draft of this work. John W. Pearson gratefully acknowledges support from the Engineering and Physical Research Council (EPSRC) grant EP/S027785/1, and a Fellowship from The Alan Turing Institute. Jennifer Pestana gratefully acknowledges support from the EPSRC grant EP/R009821/1.

References

[1] I. S. Duff, A. M. Erisman, J. K. Reid, *Direct Methods for Sparse Matrices 2nd ed.*, Oxford University Press, **2017**.

[2] S. Bellavia, J. Gondzio, B. Morini, *SIAM J. Sci. Comput.* **2013**, *35*, A192–A211.

[3] A. V. Knyazev, *SIAM J. Sci. Comput.* **2001**, *23*, 517–541.

[4] M. Benzi, *J. Comput. Phys.* **2002**, *182*, 418–477.

[5] A. J. Wathen, *Acta Numer.* **2015**, *24*, 329–376.

[6] D. Bertaccini, F. Durastante, *Iterative Methods and Preconditioning for Large and Sparse Linear Systems with Applications*, CRC Press, **2018**.

[7] K. Chen, *Matrix Preconditioning Techniques and Applications*, *Vol. 19*, Cambridge University Press, **2005**.

[8] O. Axelsson, *BIT Numer. Math.* **1985**, *25*, 166–187.

[9] M. Benzi, A. J. Wathen, in *Model Order Reduction: Theory, Research Aspects and Applications. Mathematics in Industry (The European Consortium for Mathematics in Industry), Vol. 13*, (Eds: W. H. A. Schilders, H. A. van der Vorst, J. Rommes), Springer-Verlag Berlin Heidelberg, **2008**, pp. 195–211.

[10] M. A. Olshanskii, E. E. Tyrtyshnikov, *Iterative Methods for Linear Systems: Theory and Applications*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **2014**.

[11] G. Mele, E. Ringh, D. Ek, F. Izzo, P. Upadhyaya, E. Jarlebring, *Preconditioning for Linear Systems*, Kindle Direct Publishing, **2020**.

[12] A. M. Turing, *Q. J. Mech. Appl. Math.* **1948**, *1*, 287–308.

[13] J. A. Meijerink, H. A. van der Vorst, *Math. Comp.* **1977**, *31*, 148–162.

[14] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, *Vol. 17*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **1997**.

[15] J. Liesen, Z. Strakoš, *Krylov Subspace Methods: Principles and Analysis*, Oxford University Press, **2013**.

[16] Y. Saad, *Iterative Methods for Sparse Linear Systems*, *Vol. 82*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **2003**.

[17] M. Eiermann, O. G. Ernst, *Acta Numer.* **2001**, *10*, 251–312.

[18] J. Liesen, P. Tichý, *GAMM-Mitt.* **2004**, *27*, 153–173.

[19] V. Simoncini, D. B. Szyld, *Numer. Linear Algebra Appl.* **2007**, *14*, 1–59.

[20] A. N. Krylov, *Izv. Akad. Nauk SSSR Ser. Fiz.-Mat.* **1931**, *4*, 491–539.

[21] M. R. Hestenes, E. Stiefel, *J. Res. Nat. Bur. Standards* **1952**, *49*, 409–436.

[22] C. Lanczos, *J. Res. Nat. Bur. Standards* **1950**, *45*, 255–282.

[23] C. Lanczos, *J. Res. Nat. Bur. Standards* **1952**, *49*, 33–53.

[24] J. K. Reid, in *Large Sparse Sets of Linear Equations*, (Ed.J. K. Reid), Academic Press, **1971**, pp. 231–254.

[25] M. Engeli, T. H. Ginsburg, H. Rutishauser, E. Stiefel, *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems*, Birkhäuser Basel, **1959**.

[26] C. Paige, M. Saunders, *SIAM J. Numer. Anal.* **1975**, *12*, 617–629.

[27] Y. Saad, M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **1986**, *7*, 856–869.

[28] Y. Saad, *Math. Comp.* **1981**, *37*, 105–126.

[29] C. C. Paige, M. A. Saunders, *ACM Trans. Math. Softw.* **1982**, *8*, 43–71.

[30] D. C.-L. Fong, M. Saunders, *SIAM J. Sci. Comput.* **2011**, *33*, 2950–2971.

[31] R W. Freund, N. M. Nachtigal, *Numer. Math.* **1991**, *60*, 315–339.

[32] R. Fletcher in *Numerical Analysis (Proc. 6th Biennial Dundee Conf., Univ. Dundee, 1975)*, Vol. 506 of *Lecture Notes in Mathematics*, Springer-Verlag, **1976**, pp. 73–89.

[33] H. A. van der Vorst, *SIAM J. Sci. Stat. Comput.* **1992**, *13*, 631–644.

[34] P. Wesseling, P. Sonneveld, in *Approximation Methods for Navier–Stokes Problems (Proc. Sympos., Univ. Paderborn, 1979)*, Springer-Verlag, Vol. 771 of *Lecture Notes in Mathematics*, **1980**, pp. 543–562.

[35] P. Sonneveld, M. B. van Gijzen, *SIAM J. Sci. Comput.* **2008**, *31*, 1035–1062.

[36] M. Embree, How descriptive are GMRES convergence bounds?, *Technical Report NA-99-08*, University of Oxford, **1999**.

[37] A. Greenbaum, Z. Strakoš, in *Recent Advances in Iterative Methods*, (Eds: G. Golub, M. Luskin, A. Greenbaum), Springer, New York, **1994**, pp. 95–118.

[38] M. Arioli, V. Pták, Z. Strakoš, *BIT Numer. Math.* **1998**, *38*, 636–643.

[39] A. Greenbaum, V. Pták, Z. Strakoš, *SIAM J. Matrix Anal. Appl.* **1996**, *17*, 465–469.

[40] G. Meurant, J. Duintjer Tebbens, *Numer. Alg.* **2015**, *68*, 143–165.

[41] D. Braess, P. Peisker, *IMA J. Numer. Anal.* **1986**, *6*, 393–404.

[42] M. Arioli, J. Liesen, A. Międlar, Z. Strakoš, *GAMM-Mitt.* **2013**, *36*, 102–129.

[43] J. Papež, J. Liesen, Z. Strakoš, *Linear Alg. Appl.* **2014**, *449*, 89–114.

[44] M. Arioli, E. H. Georgoulis, D. Loghin, *SIAM J. Sci. Comput.* **2013**, *35*, A1537–A1559.

[45] M. Arioli, D. Loghin, A. J. Wathen, *Numer. Math.* **2005**, *99*, 381–410.

[46] P. Jiránek, Z. Strakoš, M. Vohralík, *SIAM J. Sci. Comput.* **2010**, *32*, 1567–1590.

[47] D. N. Arnold, R. S. Falk, R. Winther, *Math. Comp.* **1997**, *66*, 957–984.

[48] V. Faber, T. A. Manteuffel, S. V. Parter, *Adv. Appl. Math.* **1990**, *11*, 109–163.

[49] A. Klawonn, Ph.D. thesis, Universität Münster, **1996**.

[50] P. Oswald, *Z. Anal. Anwendungen* **1990**, *9*, 43–64.

[51] P. Oswald, Norm equivalencies and multilevel Schwarz preconditioning for variational problems, *Forschungsergebnisse Math/92/01*, Friedrich Schiller Universität Jena, **1992**.

[52] W. Zulehner, *SIAM J. Matrix Anal. Appl.* **2011**, *32*, 536–560.

[53] U. Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **1993**.

[54] J. Málek, Z. Strakoš, *Preconditioning and the Conjugate Gradient Method in the Context of Solving PDEs*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **2015**.

[55] K.-A. Mardal, R. Winther, *Numer. Linear Algebra Appl.* **2011**, *18*, 1–40.

[56] J. Hrnčíř, I. Pultarová, Z. Strakoš, *Numer. Alg.* **2020**, *83*, 50–98.

[57] H. Yserentant, *Computing* **1985**, *35*, 39–49.

[58] H. Yserentant, *Numer. Math.* **1986**, *49*, 379–412.

[59] J. Xu, *SIAM Rev.* **1992**, *34*, 581–613.

[60] L. Ling, E. J. Kansa, *Adv. Comput. Math.* **2005**, *23*, 31–54.

[61] D. Brown, L. Ling, E. Kansa, J. Levesley, *Eng. Anal. Bound. Elem.* **2005**, *29*, 343–353.

[62] R. K. Beatson, J. B. Cherrie, C. T. Mouat, *Adv. Comput. Math.* **1999**, *11*, 253–270.

[63] R. E. Bank, *SIAM J. Numer. Anal.* **1977**, *14*, 950–970.

[64] B. L. Buzbee, G. H. Golub, C. W. Nielson, *SIAM J. Numer. Anal.* **1970**, *7*, 627–656.

[65] R. W. Hockney, *J. ACM* **1965**, *12*, 95–113.

[66] D. Fischer, G. Golub, O. Hald, C. Leiva, O. Widlund, *Math. Comp.* **1974**, *28*, 349–368.

[67] T. Gergelits, K.-A. Mardal, B. F. Nielsen, Z. Strakoš, *SIAM J. Numer. Anal.* **2019**, *57*, 1369–1394.

[68] U. Trottenberg, C. Oosterlee, A. Schüller, *Multigrid*, Academic Press, **2001**.

[69] R. P. Fedorenko, *USSR Comput. Math. Math. Phys.* **1962**, *1*, 1092–1096.

[70] R. P. Fedorenko, *USSR Comput. Math. Math. Phys.* **1964**, *4*, 227–235.

[71] A. Brandt, *Math. Comp.* **1977**, *31*, 333–390.

[72] A. Brandt, S. McCormick, J. Ruge, Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations, *Technical Report*, Institute for Computational Studies, Fort Collins, CO, **1982**.

[73] A. Brandt, S. McCormick, J. Ruge in *Sparsity and its Applications*, D. J. Evans (Ed.), Cambridge University Press, **1985**, pp. 257–284.

[74] A. Brandt, *Appl. Math. Comput.* **1986**, *19*, 23–56.

[75] J. E. Dendy, *J. Comput. Phys.* **1982**, *48*, 366–386.

[76] J. W. Ruge, K. Stüben in *Multigrid Methods*, S. F. McCormick (Ed.), Society for Industrial and Applied Mathematics, Philadelphia, PA, **1987**, pp. 73–130.

[77] P. Wesseling, C. W. Oosterlee, *J. Comput. Appl. Math.* **2001**, *128*, 311–334.

[78] J. Xu, L. Zikatanov, *Acta Numer.* **2017**, *26*, 591–721.

[79] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag Berlin Heidelberg, **1985**.

[80] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley & Sons, New York, **1992**.

[81] H. A. Schwarz, *Vierteljahrsschr. Naturf. Ges. Zürich* **1870**, *15*, 272–286.

[82] M. Dryja, O. Widlund, An additive variant of the Schwarz alternating method for the case of many subregions, *Technical Report 339*, Department of Computer Science, Courant Institute, **1987**.

[83] M. Dryja, O. B. Widlund, in *Iterative Methods for Large Linear Systems*, (Eds: D. R. Kincaid, L. J. Hayes), Academic Press, **1990**, pp. 273–291.

[84] M. Benzi, A. Frommer, R. Nabben, D. B. Szyld, *Numer. Math.* **2001**, *89*, 605–639.

[85] A. Frommer, D. B. Szyld, *SIAM J. Numer. Anal.* **2001**, *39*, 463–479.

[86] R. A. Nicolaides, *SIAM J. Numer. Anal.* **1987**, *24*, 355–365.

[87] F. Nataf, H. Xiang, V. Dolean, N. Spillane, *SIAM J. Sci. Comput.* **2011**, *33*, 1623–1642.

[88] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, *Numer. Math.* **2014**, *126*, 741–770.

[89] V. Dolean, P. Jolivet, F. Nataf, *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **2015**.

[90] M. J. Gander, *Electron. Trans. Numer. Anal.* **2008**, *31*, 228–255.

[91] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, *Vol. 95*, Springer, **1994**.

[92] T. P. A. Mathew, *Domain Decomposition Methods for the Numerical Solution of Partial Differential Equations*, Springer-Verlag Berlin Heidelberg, **2008**.

[93] A. Quarteroni, A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, **1999**.

[94] B. Smith, P. Bjørstad, W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, **1996**.

[95] A. Toselli, O. Widlund, *Domain Decomposition Methods – Algorithms and Theory*, Springer-Verlag Berlin Heidelberg, **2005**.

[96] R. E. Bank, *SIAM J. Numer. Anal.* **1981**, *18*, 724–743.

[97] H. Calandra, S. Gratton, X. Vasseur, in *Modern Solvers for Helmholtz Problems*, (Eds: D. Lahaye, J. Tang, K. Vuik), Birkhäuser, Cham, **2017**, pp. 141–155.

[98] H. C. Elman, O. G. Ernst, D. P. O'Leary, *SIAM J. Sci. Comput.* **2001**, *23*, 1291–1315.

[99] E. Haber, S. MacLachlan, *J. Comput. Phys.* **2011**, *230*, 4403–4418.

[100] M. Bollhöfer, M. J. Grote, O. Schenk, *SIAM J. Sci. Comput.* **2009**, *31*, 3781–3805.

[101] X.-C. Cai, O. B. Widlund, *SIAM J. Sci. Stat. Comput.* **1992**, *13*, 243–258.

[102] M. J. Gander, H. Zhang, in *Domain Decomposition Methods in Science and Engineering XX*, (Eds: R. Bank, M. Holst, O. Widlund, J. Xu), Springer-Verlag Berlin Heidelberg, **2013**, pp. 215–222.

[103] C. Farhat, A. Macedo, M. Lesoinne, *Numer. Math.* **2000**, *85*, 283–308.

[104] J.-D. Benamou, B. Desprès, *J. Comput. Phys.* **1997**, *136*, 68–82.

[105] M. J. Gander, F. Magoulès, F. Nataf, *SIAM J. Sci. Comput.* **2002**, *24*, 38–60.

[106] S. Kim, *Appl. Numer. Math.* **1995**, *17*, 411–429.

[107] A. Bayliss, C. I. Goldstein, E. Turkel, *J. Comput. Phys.* **1983**, *49*, 443–457.

[108] A. L. Laird, M. B. Giles, Preconditioned iterative solution of the 2D Helmholtz equation, *Technical Report 02/12*, University of Oxford, **2002**.

[109] M. Magolu monga Made, R. Beauwens, G. Warzée, *Commun. Numer. Methods Eng.* **2000**, *16*, 801–817.

[110] Y. A. Erlangga, C. W. Oosterlee, C. Vuik, *SIAM J. Sci. Comput.* **2006**, *27*, 1471–1492.

[111] A. H. Sheikh, D. Lahaye, C. Vuik, *Numer. Linear Algebra Appl.* **2013**, *20*, 645–662.

[112] P.-H. Cocquet, M. J. Gander, *SIAM J. Sci. Comput.* **2017**, *39*, A438–A478.

[113] M. J. Gander, I. G. Graham, E. A. Spence, *Numer. Math.* **2015**, *131*, 567–614.

[114] O. Axelsson, J. Karátson, F. Magoulès, *J. Comput. Appl. Math.* **2018**, *340*, 424–431.

[115] M. J. Gander, F. Nataf, *J. Comput. Acoust.* **2005**, *13*, 455–476.

[116] B. Engquist, L. Ying, *Multiscale Model. Simul.* **2011**, *9*, 686–710.

[117] M. Gander, H. Zhang, *SIAM Rev.* **2019**, *61*, 3–76.

[118] X. Liu, Y. Xi, Y. Saad, M. V. de Hoop, *SIAM J. Matrix Anal. Appl.* **2020**, *41*, 58–82.

[119] Y. A. Erlangga, *Arch. Comput. Methods Eng.* **2008**, *15*, 37–66.

[120] O. G. Ernst, M. J. Gander, in *Numerical Analysis of Multiscale Problems*, (Eds: I. G. Graham, T. Y. Hou, O. Lakkis, R. Scheichl), Springer-Verlag Berlin Heidelberg, **2012**, pp. 325–363.

[121] L. N. Trefethen, M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, Princeton, NJ, **2005**.

[122] O. Axelsson, J. Karátson, *Numer. Func. Anal. Opt.* **2003**, *24*, 455–474.

[123] H. C. Elman, M. H. Schultz, *SIAM J. Numer. Anal.* **1986**, *23*, 44–57.

[124] Z.-Z. Bai, G. H. Golub, M. K. Ng, *SIAM J. Matrix Anal. Appl.* **2003**, *24*, 603–626.

[125] D. Bertaccini, G. H. Golub, S. Serra Capizzano, C. T. Possio, *Numer. Math.* **2005**, *99*, 441–484.

[126] R. C. Y. Chin, T. A. Manteuffel, *SIAM J. Numer. Anal.* **1988**, *25*, 564–585.

[127] H. C. Elman, M. P. Chernesky, *SIAM J. Numer. Anal.* **1993**, *30*, 1268–1290.

[128] H. C. Elman, D. J. Silvester, A. J. Wathen, *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics 2nd ed.*, Oxford University Press, **2014**.

[129] H. C. Elman, G. H. Golub, *Math. Comp.* **1991**, *56*, 215–242.

[130] H. Han, V. P. Il'in, R. B. Kellogg, W. Yuan, *J. Comp. Math.* **1992**, *10*, 57–76.

[131] P. A. Farrell, *IMACS Ann. Comput. Appl. Math* **1989**, *1*, 681–686.

[132] O. Axelsson, J. Karátson, *SIAM J. Numer. Anal.* **2007**, *45*, 1495–1516.

[133] R. C. Y. Chin, T. A. Manteuffel, J. de Pillis, *SIAM J. Sci. Stat. Comput.* **1984**, *5*, 281–299.

[134] E. L. Wachspress, *J. Soc. Ind. Appl. Math.* **1963**, *11*, 994–1016.

[135] D. Palitta, V. Simoncini, *BIT Numer. Math.* **2016**, *56*, 751–776.

[136] W. Hackbusch, *Computing* **1999**, *62*, 89–108.

[137] J. Bey, G. Wittum, *Appl. Numer. Math.* **1997**, *23*, 177–192.

[138] W. Hackbusch, T. Probst, *Numer. Linear Algebra Appl.* **1997**, *4*, 85–102.

[139] W. A. Mulder, *J. Comput. Phys.* **1989**, *83*, 303–323.

[140] T. Washio, C. W. Oosterlee, *SIAM J. Sci. Comput.* **1998**, *19*, 1646–1666.

[141] A. Brandt, I. Yavneh, *J. Comput. Phys.* **1992**, *101*, 151–164.

[142] A. Ramage, *J. Comput. Appl. Math.* **1999**, *110*, 187–203.

[143] M. Benzi, M. A. Olshanskii, *SIAM J. Sci. Comput.* **2006**, *28*, 2095–2113.

[144] P. E. Farrell, L. Mitchell, F. Wechsung, *SIAM J. Sci. Comput.* **2019**, *41*, A3073–A3096.

[145] J. Boyle, M. Mihajlović, J. Scott, *Int. J. Numer. Meth. Eng.* **2010**, *82*, 64–98.

[146] Y. Notay, *SIAM J. Sci. Comput.* **2012**, *34*, A2288–A2316.

[147] C.-T. Wu, H. C. Elman, *SIAM J. Sci. Comput.* **2006**, *28*, 2208–2228.

[148] V. Dolean, F. Nataf, G. Rapin, *Math. Comp.* **2009**, *78*, 789–814.

[149] J. Pestana, R. Muddle, M. Heil, F. Tisseur, M. Mihajlović, *SIAM J. Sci. Comput.* **2016**, *38*, A325–A345.

[150] P. Farrell, J. Pestana, *Numer. Linear Algebra Appl.* **2015**, *22*, 731–747.

[151] M. Benzi, G. H. Golub, J. Liesen, *Acta Numer.* **2005**, *14*, 1–137.

[152] M. Benzi, M. Ng, Q. Niu, Z. Wang, *J. Comput. Phys.* **2011**, *230*, 6185–6202.

[153] M. Benzi, S. Deparis, G. Grandperrin, A. Quarteroni, *Comput. Meth. Appl. Mech. and Engrg.* **2016**, *300*, 129–145.

[154] M. F. Murphy, G. H. Golub, A. J. Wathen, *SIAM J. Sci. Comput.* **2000**, *21*, 1969–1972.

[155] N. Bootland, A. Bentley, C. Kees, A. Wathen, *SIAM J. Sci. Comput.* **2019**, *41*, B843–B869.

[156] H. C. Elman, *SIAM J. Sci. Comput.* **1999**, *20*, 1299–1316.

[157] H. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro, *SIAM J. Sci. Comput.* **2006**, *27*, 1651–1668.

[158] T. Roy, T. B. Jönsthövel, C. Lemon, A. J. Wathen, *J. Comput. Phys.* **2019**, *395*, 636–652.

[159] J. A. White, R. I. Borja, *Comput. Geosci.* **2011**, *15*, 647–659.

[160] Yu. A. Kuznetsov, *Russ. J. Numer. Anal. Math. Modelling* **1995**, *10*, 187–211.

[161] G. H. Golub, R. S. Varga, *Numer. Math.* **1961**, *3*, 147–156.

[162] G. H. Golub, R. S. Varga, *Numer. Math.* **1961**, *3*, 157–168.

[163] A. Wathen, T. Rees, *Electron. Trans. Numer. Anal.* **2009**, *34*, 125–135.

[164] F. P. Ali Beik, M. Benzi, *SIAM J. Matrix Anal. Appl.* **2018**, *39*, 902–921.

[165] G. N. Gatica, N. Heuer, *SIAM J. Numer. Anal.* **2000**, *38*, 380–400.

[166] K.-A. Mardal, B. F. Nielsen, M. Nordaas, *BIT Numer. Math.* **2017**, *57*, 405–431.

[167] J. Sogn, W. Zulehner, *IMA J. Numer. Anal.* **2019**, *39*, 1328–1359.

[168] C. Powell, D. Silvester, in *Challenges in Scientific Computing – CISC 2002*, (Ed.E. Bänsch), Springer-Verlag Berlin Heidelberg, **2003**, pp. 268–285.

[169] J. Sundnes, G. T. Lines, K. A. Mardal, A. Tveito, *Comput. Methods Biomech. Biomed. Engin.* **2002**, *5*, 397–409.

[170] D. Silvester, A. Wathen, *SIAM J. Numer. Anal.* **1994**, *31*, 1352–1367.

[171] P. S. Vassilevski, U. Villa, *SIAM J. Sci. Comput.* **2013**, *35*, S3–S17.

[172] B. Fischer, A. Ramage, D. J. Silvester, A. J. Wathen, *BIT Numer. Math.* **1998**, *38*, 527–543.

[173] J. W. Pearson, J. Pestana, D. J. Silvester, *Numer. Math.* **2018**, *138*, 331–363.

[174] R. Herzog, *arXiv e-prints* **2020**, arXiv:2003.09478.

[175] P. P. Grinevich, M. A. Olshanskii, *SIAM J. Sci. Comput.* **2009**, *31*, 3959–3978.

[176] S. Holmgren, K. Otto, *SIAM J. Sci. Comput.* **1994**, *15*, 385–407.

[177] K. Otto, *SIAM J. Numer. Anal.* **1996**, *33*, 2131–2165.

[178] M. M. Meerschaert, C. Tadjeran, *J. Comput. Appl. Math.* **2004**, *172*, 65–77.

[179] M. M. Meerschaert, C. Tadjeran, *Appl. Numer. Math.* **2006**, *56*, 80–90.

[180] G. H. Golub, C. F. van Loan, *Matrix Computations 4th ed.*, The John Hopkins University Press, Maryland, USA, **2013**.

[181] U. Grenander, G. Szegő, *Toeplitz Forms and their Applications*, American Mathematical Society, **1958**.

[182] E. E. Tyrtyshnikov, *Linear Alg. Appl.* **1996**, *232*, 1–43.

[183] N. L. Zamarashkin, E. E. Tyrtyshnikov, *Sb. Math.* **1997**, *188*, 1191–1201.

[184] P. Tilli, *Linear Multilinear Algebra* **1998**, *45*, 147–159.

[185] G. Strang, *Stud. Appl. Math.* **1986**, *74*, 171–176.

[186] J. A. Olkin, Ph.D. thesis, Rice University, **1986**.

[187] T. F. Chan, *SIAM J. Sci. Stat. Comput.* **1988**, *9*, 766–771.

[188] E. E. Tyrtyshnikov, *SIAM J. Matrix Anal. Appl.* **1992**, *13*, 459–473.

[189] D. Bini, P. Favati, *SIAM J. Matrix Anal. Appl.* **1993**, *14*, 500–507.

[190] T. Huckle, *BIT Numer. Math.* **1994**, *34*, 99–112.

[191] R. H. Chan, *IMA J. Numer. Anal.* **1991**, *11*, 333–345.

[192] R. H. Chan, P. T. P. Tang, *SIAM J. Sci. Comput.* **1994**, *15*, 164–171.

[193] G. Fiorentino, S. Serra, *Calcolo* **1991**, *28*, 283–305.

[194] R. H. Chan, M. K. Ng, *SIAM Rev.* **1996**, *38*, 427–482.

[195] R. H.-F. Chan, X.-Q. Jin, *An Introduction to Iterative Toeplitz Solvers*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **2007**.

[196] M. K. Ng, *Iterative Methods for Toeplitz Systems*, Oxford University Press, **2004**.

[197] S. Serra Capizzano, E. Tyrtyshnikov, *Math. Comp.* **2003**, *72*, 1305–1316.

[198] S. Serra Capizzano, E. Tyrtyshnikov, *SIAM J. Matrix Anal. Appl.* **1999**, *21*, 431–439.

[199] A. Aricò, M. Donatelli, *Numer. Math.* **2007**, *105*, 511–547.

[200] S. Serra, *BIT Numer. Math.* **1999**, *39*, 152–175.

[201] C. Garoni, S. Serra-Capizzano, *Generalized Locally Toeplitz Sequences: Theory and Applications*, *Vol. I*, Springer, **2017**.

[202] C. Garoni, S. Serra-Capizzano, *Generalized Locally Toeplitz Sequences: Theory and Applications*, *Vol. II*, Springer, **2018**.

[203] M. J. Gander, in *Multiple Shooting and Time Domain Decomposition Methods*, (Eds: T. Carraro, M. Geiger, S. Körkel, R. Rannacher), Springer, **2015**, pp. 69–113.

[204] D. Palitta, *arXiv e-prints* **2019**, arXiv:1908.11851.

[205] E. McDonald, J. Pestana, A. Wathen, *SIAM J. Sci. Comput.* **2018**, *40*, A1012–A1033.

[206] A. Goddard, A. Wathen, *Electron. Trans. Numer. Anal.* **2019**, *51*, 135–150.

[207] S. Güttel, J. W. Pearson, *IMA J. Numer. Anal.* **2018**, *38*, 1861–1892.

[208] M. Benzi, E. Haber, L. Taralli, *Adv. Comput. Math.* **2011**, *35*, 149–173.

[209] M. Hinze, M. Köster, S. Turek in *Constrained Optimization and Optimal Control for Partial Differential Equations*, Vol. 160 of *Internat. Ser. Numer. Math.*, Springer Basel AG, **2012**, pp. 147–170.

[210] M. Stoll, A. Wathen, All-at-once solution of time-dependent PDE-constrained optimization problems, *Technical Report NA-10-13*, University of Oxford, **2010**.

[211] I. G. Graham, M. J. Hagger, *SIAM J. Sci. Comput.* **1999**, *20*, 2041–2066.

[212] P. Bastian, M. Blatt, R. Scheichl, *Numer. Linear Algebra Appl.* **2012**, *19*, 367–388.

[213] J. Kraus, R. Lazarov, M. Lymbery, S. Margenov, L. Zikatanov, *SIAM J. Sci. Comput.* **2016**, *38*, A875–A898.

[214] A. Klawonn, O. B. Widlund, M. Dryja, *SIAM J. Numer. Anal.* **2002**, *40*, 159–179.

[215] C. E. Powell, D. Silvester, *SIAM J. Matrix Anal. Appl.* **2004**, *25*, 718–738.

[216] M. W. Benson, Master's thesis, Lakehead University, Thunder Bay, Ontario, **1973**.

[217] G. Alléon, M. Benzi, L. Giraud, *Numer. Alg.* **1997**, *16*, 1–15.

[218] B. Carpentieri, I. S. Duff, L. Giraud, *Numer. Linear Algebra Appl.* **2000**, *7*, 667–685.

[219] A. Franceschini, M. Ferronato, C. Janna, V. A. P. Magri, in *Seminari Padovani di Analisi Numerica*, (Eds: E. Facca, A. Martínez, E. Perracchione, F. Piazzon), **2018**, pp. 11–22.

[220] L. Giraud, J. Langou, G. Sylvand, *J. Comput. Acoust.* **2006**, *14*, 1–29.

[221] L. Yu. Kolotilina, A. Yu. Yeremin, *SIAM J. Matrix Anal. Appl.* **1993**, *14*, 45–58.

[222] M. Margonari, M. Bonnet, *Comput. Struct.* **2005**, *83*, 700–717.

[223] J. A. Sanz, M. Bonnet, J. Dominguez, *Eng. Anal. Bound. Elem.* **2008**, *32*, 787–795.

[224] M. J. Grote, T. Huckle, *SIAM J. Sci. Comput.* **1997**, *18*, 838–853.

[225] M. Benzi, M. Tůma, *SIAM J. Sci. Comput.* **1998**, *19*, 968–994.

[226] M. Benzi, M. Tůma, *Appl. Numer. Math.* **1999**, *30*, 305–340.

[227] T. Huckle, *Appl. Numer. Math.* **1999**, *30*, 291–303.

[228] T. K. Huckle, *Numer. Linear Algebra Appl.* **1998**, *5*, 57–71.

[229] C. H. Ahn, B. Carpentieri, I. S. Duff, L. Giraud, *Electromagnetics* **1999**, *19*, 131–146.

[230] B. Carpentieri, I. S. Duff, L. Giraud, M. Magolu monga Made, *Numer. Linear Algebra Appl.* **2004**, *11*, 753–771.

[231] M. Wathen, C. Greif, *SIAM J. Sci. Comput.* **2020**, *42*, B57–B79.

[232] D. W. Peaceman, H. H. Rachford, Jr., *J. Soc. Ind. Appl. Math.* **1955**, *3*, 28–41.

[233] Z. Dostál, *Int. J. Comput. Math.* **1988**, *23*, 315–323.

[234] A. Gaul, M. H. Gutknecht, J. Liesen, R. Nabben, *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 495–518.

[235] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Springer-Verlag Berlin Heidelberg, **2008**.

[236] L. Grasedyck, W. Hackbusch, *Computing* **2003**, *70*, 295–334.

[237] W. Hackbusch, B. Khoromskij, S. A. Sauter, in *Lectures on Applied Mathematics*, (Eds: H.-J. Bungartz, Ronald H. W. Hoppe, C. Zenger), Springer-Verlag Berlin Heidelberg, **2000**, pp. 9–29.

[238] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, D. White, *SIAM J. Matrix Anal. Appl.* **2005**, *27*, 341–364.

[239] L. Greengard, V. Rokhlin, *J. Comput. Phys.* **1997**, *135*, 280–292.

[240] K. L. Ho, L. Greengard, *SIAM J. Sci. Comput.* **2012**, *34*, A2507–A2532.

[241] S. Le Borne, L. Grasedyck, *SIAM J. Matrix Anal. Appl.* **2006**, *27*, 1172–1183.

[242] H. Ibeid, R. Yokota, J. Pestana, D. Keyes, *Comput. Vis. Sci.* **2018**, *18*, 213–229.

[243] R. Bridson, C. Greif, *SIAM J. Matrix Anal. Appl.* **2006**, *27*, 1056–1068.

[244] C. Greif, T. Rees, D. B. Szyld, *SeMA Journal* **2017**, *74*, 213–231.

[245] T. Bakhos, P. K. Kitanidis, S. Ladenheim, A. K. Saibaba, D. B. Szyld, *SIAM J. Sci. Comput.* **2017**, *39*, S222–S247.

[246] N. Bootland, A. Wathen, *arXiv e-prints* **2020**, arXiv:2005.07608.

[247] J. Gondzio, *Eur. J. Oper. Res.* **2012**, *218*, 587–601.

[248] L. Bergamaschi, J. Gondzio, G. Zilli, *Comput. Optim. Appl.* **2004**, *28*, 149–171.

[249] P. E. Gill, W. Murray, D. B. Ponceleón, M. A. Saunders, *SIAM J. Matrix Anal. Appl.* **1992**, *13*, 292–311.

[250] A. R. L. Oliveira, D. C. Sorensen, *Linear Alg. Appl.* **2005**, *394*, 1–24.

[251] L. Schork, J. Gondzio, Implementation of an interior point method with basis preconditioning, *Technical Report ERGO 18-014*, The University of Edinburgh, **2018**.

[252] T. A. Davis, *ACM Trans. Math. Softw.* **2004**, *30*, 196–199.

[253] X. S. Li, *ACM Trans. Math. Softw.* **2005**, *31*, 302–325.

[254] M. Bollhöfer, *Linear Alg. Appl.* **2001**, *338*, 201–218.

[255] E. Chow, Y. Saad, *J. Comput. Appl. Math.* **1997**, *86*, 387–414.

[256] M. T. Jones, P. E. Plassmann, *ACM Trans. Math. Softw.* **1995**, *21*, 5–17.

[257] Y. Notay, *Appl. Numer. Math.* **1999**, *31*, 209–225.

[258] Y. Saad, *SIAM J. Sci. Comput.* **1996**, *17*, 830–847.

[259] S. C. Eisenstat, *SIAM J. Sci. Stat. Comput.* **1981**, *2*, 1–4.

[260] M. Benzi, M. Tůma, *Numer. Linear Algebra Appl.* **2003**, *10*, 385–400.

[261] R. E. Bank, C. Wagner, *Numer. Math.* **1999**, *82*, 543–576.

[262] P. S. Vassilevski, *Multilevel Block Factorization Preconditioners: Matrix-Based Analysis and Algorithms for Solving Finite Element Equations*, Springer-Verlag New York, **2008**.

[263] J.-S. Chai, K.-C. Toh, *Comput. Optim. Appl.* **2007**, *36*, 221–247.

[264] J. C. Haws, C. D. Meyer, Preconditioning KKT systems, *Technical Report M&CT-Tech-01-021*, The Boeing Company, **2003**.

[265] O. Schenk, A. Wächter, M. Weiser, *SIAM J. Sci. Comput.* **2008**, *31*, 939–960.

[266] N. I. M. Gould, M. E. Hribar, J. Nocedal, *SIAM J. Sci. Comput.* **2001**, *23*, 1376–1395.

[267] C. Keller, N. I. M. Gould, A. J. Wathen, *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1300–1317.

[268] L. Lukšan, J. Vlček, *Numer. Linear Algebra Appl.* **1998**, *5*, 219–247.

[269] N. Gould, D. Orban, T. Rees, *SIAM J. Matrix Anal. Appl.* **2014**, *35*, 1329–1343.

[270] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, A. J. Wathen, *SIAM J. Matrix Anal. Appl.* **2006**, *28*, 170–189.

[271] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, A. J. Wathen, *Comput. Optim. Appl.* **2007**, *36*, 249–270.

[272] T. Rees, J. Scott, *Numer. Linear Algebra Appl.* **2018**, *25*, e2103.

[273] C. Durazzi, V. Ruggiero, *Numer. Linear Algebra Appl.* **2003**, *10*, 673–688.

[274] L. Bergamaschi, J. Gondzio, M. Venturin, G. Zilli, *Comput. Optim. Appl.* **2007**, *36*, 137–147.

[275] N. Dyn, W. E. Ferguson, Jr., *Math. Comp.* **1983**, *41*, 165–170.

[276] A. Forsgren, P. E. Gill, J. D. Griffin, *SIAM J. Optim.* **2007**, *18*, 666–690.

[277] L. Lukšan, C. Matonoha, J. Vlček, *Numer. Linear Algebra Appl.* **2004**, *11*, 431–453.

[278] M. Rozložník, V. Simoncini, *SIAM J. Matrix Anal. Appl.* **2002**, *24*, 368–391.

[279] E. de Sturler, J. Liesen, *SIAM J. Sci. Comput.* **2005**, *26*, 1598–1619.

[280] J. Scott, M. Tůma, *Numer. Linear Algebra Appl.* **2017**, *24*, e2099.

[281] T. Rees, H. S. Dollar, A. J. Wathen, *SIAM J. Sci. Comput.* **2010**, *32*, 271–298.

[282] M. Kočvara, D. Loghin, J. Turner, *SIAM J. Sci. Comput.* **2016**, *38*, A128–A145.

[283] C. Greif, D. Schötzau, *Electron. Trans. Numer. Anal.* **2006**, *22*, 114–121.

[284] C. Greif, D. Schötzau, *Numer. Linear Algebra Appl.* **2007**, *14*, 281–297.

[285] G. H. Golub, C. Greif, *SIAM J. Sci. Comput.* **2003**, *24*, 2076–2092.

[286] R. Fletcher, *Practical Methods of Optimization 2nd ed.*, John Wiley & Sons, Chichester, **1987**.

[287] M. Fortin, R. Glowinski, *Augmented Lagrangian Methods: Applications to the Solution of Numerical Boundary-Value Problems*, Stud. Math. Appl., Vol. 15, North-Holland, Amsterdam, **1983**.

[288] T. Rees, C. Greif, *SIAM J. Sci. Comput.* **2007**, *29*, 1992–2007.

[289] B. Morini, V. Simoncini, M. Tani, *Numer. Linear Algebra Appl.* **2016**, *23*, 776–800.

[290] Z.-H. Cao, *Numer. Linear Algebra Appl.* **2008**, *15*, 515–533.

[291] S.-Q. Shen, T.-Z. Huang, J.-S. Zhang, *SIAM J. Matrix Anal. Appl.* **2012**, *33*, 721–741.

[292] L. Bergamaschi, R. Bru, A. Martínez, M. Putti, *Electron. Trans. Numer. Anal.* **2006**, *23*, 76–87.

[293] J. M. Martínez, *Math. Comp.* **1993**, *60*, 681–698.

[294] J. M. Martínez, *J. Comput. Appl. Math.* **1995**, *60*, 115–125.

[295] J. L. Morales, J. Nocedal, *SIAM J. Optim.* **2000**, *10*, 1079–1096.

[296] S. G. Nash, *SIAM J. Numer. Anal.* **1984**, *21*, 770–788.

[297] D. P. O'Leary, A. Yeremin, *Linear Alg. Appl.* **1994**, *212–213*, 153–168.

[298] L. Bergamaschi, R. Bru, A. Martínez, *Math. Comput. Model.* **2011**, *54*, 1863–1873.

[299] J. Duintjer Tebbens, M. Tůma, *SIAM J. Sci. Comput.* **2007**, *29*, 1918–1941.

[300] J. Duintjer Tebbens, M. Tůma, *Numer. Linear Algebra Appl.* **2010**, *17*, 997–1019.

[301] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, *SIAM J. Optim.* **2015**, *25*, 1787–1808.

[302] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, *Comput. Optim. Appl.* **2016**, *65*, 339–360.

[303] L. Bergamaschi, V. De Simone, D. di Serafino, A. Martínez, *Numer. Linear Algebra Appl.* **2018**, *25*, e2144.

[304] S. Gratton, A. Sartenaer, J. Tshimanga, *SIAM J. Optim.* **2011**, *21*, 912–935.

[305] M. Fisher, S. Gratton, S. Gürol, Y. Trémolet, X. Vasseur, *Optim. Method. Softw.* **2017**, *33*, 45–69.

[306] S. Bellavia, V. De Simone, D. di Serafino, B. Morini, *SIAM J. Sci. Comput.* **2011**, *33*, 1785–1809.

[307] M. Benzi, D. Bertaccini, *BIT Numer. Math.* **2003**, *43*, 231–244.

[308] X.-M. Gu, T.-Z. Huang, G. Yin, B. Carpentieri, C. Wen, L. Du, *J. Comput. Appl. Math.* **2018**, *331*, 166–177.

[309] L. Bergamaschi, *Algorithms* **2020**, *13*, 100.

[310] R. Bru, J. Marín, J. Mas, M. Tůma, *SIAM J. Sci. Comput.* **2014**, *36*, A2002–A2022.

[311] N. Gould, J. Scott, *ACM Trans. Math. Softw.* **2017**, *43*, Art. 36.

[312] J. Scott, M. Tůma, *ACM Trans. Math. Softw.* **2014**, *40*, Art. 24.

[313] J. Scott, M. Tůma, *SIAM J. Sci. Comput.* **2014**, *36*, A609–A633.

[314] J. Scott, *SIAM J. Sci. Comput.* **2017**, *39*, C319–C339.

[315] Z.-Z. Bai, I. S. Duff, A. J. Wathen, *BIT Numer. Math.* **2001**, *41*, 53–70.

[316] A. T. Papadopoulus, I. S. Duff, A. J. Wathen, *BIT Numer. Math.* **2005**, *45*, 159–179.

[317] A. Jennings, M. A. Ajiz, *SIAM J. Sci. Stat. Comput.* **1984**, *5*, 978–987.

[318] X. Wang, K. A. Gallivan, R. Bramley, *SIAM J. Sci. Comput.* **1997**, *18*, 516–536.

[319] N. Li, Y. Saad, *SIAM J. Matrix Anal. Appl.* **2006**, *28*, 524–550.

[320] M. Benzi, M. Tůma, *SIAM J. Sci. Comput.* **2003**, *25*, 499–512.

[321] J. Scott, M. Tůma, *SIAM J. Sci. Comput.* **2014**, *36*, A2984–A3010.

[322] C. Greif, S. He, P. Liu, *ACM Trans. Math. Softw.* **2017**, *44*, Art. 1.

[323] X. Cui, K. Hayami, *Jpn. J. Ind. Appl. Math.* **2009**, *26*, 1–14.

[324] X. Cui, K. Hayami, J.-F. Yin, *Adv. Comput. Math.* **2011**, *35*, 243–269.

[325] A. Björck, *Numerical Methods for Least Squares Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, **1996**.

[326] A. Björck, J. Y. Yuan, *Electron. Trans. Numer. Anal.* **1999**, *8*, 26–35.

[327] M. Arioli, I. S. Duff, *SIAM J. Sci. Comput.* **2015**, *37*, S544–S561.

[328] M. Benzi, M. K. Ng, *SIAM J. Matrix Anal. Appl.* **2006**, *27*, 1106–1124.

[329] S. Börm, S. Le Borne, *Int. J. Numer. Meth. Fl.* **2012**, *68*, 83–98.

[330] Y. A. Erlangga, C. Vuik, C. W. Oosterlee, *Appl. Numer. Math.* **2006**, *56*, 648–666.

[331] Z.-Z. Bai, M. K. Ng, Z.-Q. Wang, *SIAM J. Matrix Anal. Appl.* **2009**, *31*, 410–433.

[332] P. Chidyagwai, S. Ladenheim, D. B. Szyld, *SIAM J. Sci. Comput.* **2016**, *38*, A668–A690.

[333] I. Perugia, V. Simoncini, *Numer. Linear Algebra Appl.* **2000**, *7*, 585–616.

[334] M. Benzi, M. A. Olshanskii, Z. Wang, *Int. J. Numer. Meth. Fl.* **2011**, *66*, 486–508.

[335] J. W. Pearson, A. J. Wathen, *Numer. Linear Algebra Appl.* **2012**, *19*, 816–829.

[336] J. Schöberl, W. Zulehner, *SIAM J. Matrix Anal. Appl.* **2007**, *29*, 752–773.

[337] J. Pearson, Ph.D. thesis, University of Oxford, **2013**.

[338] T. Rees, M. Stoll, *Numer. Linear Algebra Appl.* **2010**, *17*, 977–996.

[339] W. Krendl, V. Simoncini, W. Zulehner, *Numer. Math.* **2013**, *124*, 183–213.

[340] J. W. Pearson, A. J. Wathen, *Electron. Trans. Numer Anal.* **2013**, *40*, 294–310.

[341] J. W. Pearson, M. Stoll, A. J. Wathen, *SIAM J. Matrix Anal. Appl.* **2012**, *33*, 1126–1152.

[342] Z.-Z. Bai, M. Benzi, F. Chen, Z.-Q. Wang, *IMA J. Numer. Anal.* **2013**, *33*, 343–369.

[343] O. Axelsson, S. Farouq, M. Neytcheva, *Numer. Alg.* **2016**, *73*, 631–663.

[344] R. Herzog, E. Sachs, *SIAM J. Matrix Anal. Appl.* **2010**, *31*, 2291–2317.

[345] R. Herzog, S. Mach, *SIAM J. Numer. Anal.* **2016**, *54*, 688–718.

[346] J. W. Pearson, J. Gondzio, *Numer. Math.* **2017**, *137*, 959–999.

[347] J. W. Pearson, M. Stoll, A. J. Wathen, *Numer. Linear Algebra Appl.* **2012**, *21*, 81–97.

[348] M. Porcelli, V. Simoncini, M. Tani, *SIAM J. Sci. Comput.* **2015**, *37*, S472–S502.

[349] E. Haber, U. M. Ascher, *Inverse Probl.* **2001**, *17*, 1847–1864.

[350] J. Berns-Müller, I. G. Graham, A. Spence, *Linear Alg. Appl.* **2006**, *416*, 389–413.

[351] G. H. Golub, Q. Ye, *SIAM J. Sci. Comput.* **1999**, *21*, 1305–1320.

[352] M. A. Freitag, A. Spence, *Electron. Trans. Numer. Anal.* **2007**, *28*, 40–64.

[353] M. A. Freitag, A. Spence, *BIT Numer. Math.* **2007**, *47*, 27–44.

[354] V. Simoncini, L. Eldén, *BIT Numer. Math.* **2002**, *42*, 159–182.

[355] M. A. Freitag, A. Spence, *IMA J. Numer. Anal.* **2008**, *28*, 522–551.

[356] D. B. Szyld, F. Xue, *SIAM J. Matrix Anal. Appl.* **2011**, *32*, 993–1018.

[357] M. Robbé, M. Sadkane, A. Spence, *SIAM J. Matrix Anal. Appl.* **2009**, *31*, 92–113.

[358] M. A. Freitag, A. Spence, *SIAM J. Matrix Anal. Appl.* **2009**, *31*, 942–969.

[359] F. Xue, H. C. Elman, *SIAM J. Matrix Anal. Appl.* **2012**, *33*, 433–459.

[360] G. H. Golub, Q. Ye, *SIAM J. Sci. Comput.* **2002**, *24*, 312–334.

[361] R. B. Morgan, D. S. Scott, *SIAM J. Sci. Comput.* **1993**, *14*, 585–593.

[362] L. Wu, F. Xue, A. Stathopoulos, *SIAM J. Sci. Comput.* **2019**, *41*, A1013–A1040.

[363] O. Taussky, *Linear Alg. Appl.* **1972**, *5*, 147–154.

[364] S. Hon, A. Wathen, *Numer. Alg.* **2018**, *79*, 1211–1230.

[365] J. Pestana, A. J. Wathen, *SIAM J. Matrix Anal. Appl.* **2015**, *36*, 273–288.

[366] J. Pestana, *SIAM J. Matrix Anal. Appl.* **2019**, *40*, 870–887.

[367] P. Ferrari, I. Furci, S. Hon, M. A. Mursaleen, S. Serra-Capizzano, *SIAM J. Matrix Anal. Appl.* **2019**, *40*, 1066–1086.

[368] M. Mazza, J. Pestana, *BIT Numer. Math.* **2019**, *59*, 463–482.

[369] J. Pestana, Ph.D. thesis, University of Oxford, **2011**.

[370] A. Günnel, R. Herzog, E. Sachs, *Electron. Trans. Numer. Anal* **2014**, *41*, 13–20.

[371] J. H. Bramble, J. E. Pasciak, *Math. Comp.* **1988**, *50*, 1–17.

[372] P. Krzyżanowski, *Numer. Linear Algebra Appl.* **2011**, *18*, 123–140.

[373] P. Concus, G. H. Golub in *Computing Methods in Applied Sciences and Engineering*, Springer, **1976**, pp. 56–65.

[374] O. Widlund, *SIAM J. Numer. Anal.* **1978**, *15*, 801–812.

[375] D. Calvetti, B. Lewis, L. Reichel, *Numer. Math.* **2002**, *91*, 605–625.

[376] M. Hanke, *BIT Numer. Math.* **2001**, *41*, 1008–1018.

[377] T. K. Jensen, P. C. Hansen, *BIT Numer. Math.* **2007**, *47*, 103–120.

[378] M. Hanke, J. G. Nagy, R. J. Plemmons, in *Numerical Linear Algebra*, (Eds: L. Reichel, A. Ruttan, R. S. Varga), de Gruyter, Berlin, **1993**, pp. 141–163.

[379] M. Hanke, J. Nagy, *Linear Alg. Appl.* **1998**, *284*, 137–156.

[380] J. G. Nagy, K. M. Palmer, *BIT Numer. Math.* **2003**, *43*, 1003–1017.

[381] L. Eldén, V. Simoncini, *SIAM J. Matrix Anal. Appl.* **2012**, *33*, 1369–1394.

[382] Z. Ranjbar, L. Eldén, *SIAM J. Sci. Comput.* **2014**, *36*, B868–B886.

[383] J. Baglama, L. Reichel, *J. Comput. Appl. Math.* **2007**, *198*, 332–343.

[384] S. Gazzola, J. G. Nagy, *SIAM J. Sci. Comput.* **2014**, *36*, B225–B247.

[385] P. C. Hansen, T. K. Jensen, *SIAM J. Matrix Anal. Appl.* **2006**, *29*, 1–14.

[386] D. Calvetti, *J. Comput. Appl. Math.* **2007**, *198*, 378–395.

[387] V. Rokhlin, M. Tygert, *Proc. Natl. Acad. Sci. USA* **2008**, *105*, 13212–13217.

[388] P. Drineas, M. W. Mahoney, S. Muthukrishnan, T. Sarlós, *Numer. Math.* **2011**, *117*, 219–249.

[389] H. Avron, P. Maymounkov, S. Toledo, *SIAM J. Sci. Comput.* **2010**, *32*, 1217–1236.

[390] E. S. Coakley, V. Rokhlin, M. Tygert, *SIAM J. Sci. Comput.* **2011**, *33*, 849–868.

[391] X. Meng, M. A. Saunders, M. W. Mahoney, *SIAM J. Sci. Comput.* **2014**, *36*, C95–C118.

[392] J. Yang, Y.-L. Chow, C. Ré, M. W. Mahoney, *J. Mach. Learn. Res.* **2018**, *18*, 1–43.

[393] D. Calvetti, E. Somersalo, *Inverse Probl.* **2005**, *21*, 1397–1418.

[394] D. Calvetti, D. McGivney, E. Somersalo, *Inverse Probl.* **2012**, *28*, 055015.

[395] P. Hennig, *SIAM J. Optim.* **2015**, *25*, 234–260.

[396] J. Cockayne, C. J. Oates, I. C. F. Ipsen, M. Girolami, *Bayesian Anal.* **2019**, *14*, 937–1012.

[397] S. Bartels, J. Cockayne, I. C. F. Ipsen, P. Hennig, *Stat. Comput.* **2019**, doi:10.1007/s11222–019–09897–7.

[398] L. Grasedyck, *SIAM J. Matrix Anal. Appl.* **2010**, *31*, 2029–2054.

[399] W. Hackbusch, S. Kühn, *J. Fourier Anal. Appl.* **2009**, *15*, 706–722.

[400] I. V. Oseledets, *SIAM J. Sci. Comput.* **2011**, *33*, 2295–2317.

[401] I. V. Oseledets, *SIAM J. Matrix Anal. Appl.* **2010**, *31*, 2130–2145.

[402] E. E. Tyrtyshnikov, *Sbornik: Math.* **2003**, *194*, 941–954.

[403] D. Kressner, C. Tobler, *SIAM J. Matrix Anal. Appl.* **2010**, *31*, 1688–1714.

[404] D. Palitta, V. Simoncini, *Vietnam J. Math.* **2020**.

[405] R. Andreev, C. Tobler, *Numer. Linear Alg. Appl.* **2015**, *22*, 317–337.

[406] T. Breiten, V. Simoncini, M. Stoll, *Electron. Trans. Numer. Anal.* **2016**, *45*, 107–132.

[407] S. V. Dolgov, *Russ. J. Numer. Anal. M.* **2013**, *28*, 149–172.

[408] B. N. Khoromskij, *Constr. Approx.* **2009**, *30*, 599–620.

[409] S. Dolgov, M. Stoll, *SIAM J. Sci. Comput.* **2017**, *39*, A255–A280.

[410] M. Stoll, T. Breiten, *SIAM J. Sci. Comput.* **2015**, *37*, B1–B29.

[411] S. Dolgov, J. W. Pearson, D. V. Savostyanov, M. Stoll, *Appl. Math. Comput.* **2016**, *273*, 604–623.

[412] G. Heidel, V. Khoromskaia, B. N. Khoromskij, V. Schulz, *arXiv e-prints* **2020**, arXiv:1809.01971.

[413] E. C. Carson, M. Rozložník, Z. Strakoš, P. Tichý, M. Tůma, *SIAM J. Sci. Comput.* **2018**, *40*, A3549–A3580.

[414] J. Dongarra, M. A. Heroux, P. Luszczek, *Int. J. High Perform. Comput. Appl.* **2016**, *30*, 3–10.

[415] M. Hoemmen, Ph.D. thesis, UC Berkeley, **2010**.

[416] H. Anzt, J. Dongarra, M. Kreutzer, G. Wellein, M. Köhler, in *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, **2016**, pp. 683–691.

[417] V. Eijkhout, *Introduction to High Performance Scientific Computing 2nd ed.*, Lulu, **2015**.

[418] B. Bergen, T. Gradl, F. Hulsemann, U. Rude, *Comput. Sci. Eng.* **2006**, *8*, 56–62.

[419] V. E. Henson, U. M. Yang, *Appl. Numer. Math.* **2002**, *41*, 155–177.

[420] E. Chow, *Int. J. High Perform. Comput. Appl.* **2001**, *15*, 56–74.

[421] C. Janna, M. Ferronato, F. Sartoretto, G. Gambolati, *ACM Trans. Math. Softw.* **2015**, *41*, Art. 10.

[422] S. Badia, A. F. Martín, J. Principe, *SIAM J. Sci. Comput.* **2016**, *38*, C22–C52.

[423] V. A. P. Magri, A. Franceschini, M. Ferronato, C. Janna, *Numer. Linear Algebra Appl.* **2018**, *25*, e2183.

[424] M. Bernaschi, M. Carrozzo, A. Franceschini, C. Janna, *SIAM J. Sci. Comput.* **2019**, *41*, C139–C160.

[425] E. Chow, A. Patel, *SIAM J. Sci. Comput.* **2015**, *37*, C169–C193.

[426] H. Anzt, T. K. Huckle, T. Bräckle, J. Dongarra, *Parallel Computing* **2018**, *71*, 1–22.

[427] H. Ibeid, R. Yokota, D. Keyes, *arXiv e-prints* **2016**, arXiv:1608.02461.

How to cite this article: J. W. Pearson and J. Pestana (2020), Preconditioners for Krylov Subspace Methods: An Overview, *GAMM-Mitt.*, *submitted*.